**Predicting Damage and Injuries by Aircraft-Wildlife Strikes**

Paul Tuck

Introduction

Wildlife strikes against vehicles are an inevitability, and aircrafts are no exception. Though most strikes do not do serious harm to the aircraft, they do have the potential to cause serious damage, and even cause injury or death to the pilots and passengers on board. Being able to determine what the biggest risk factors are in wildlife strikes is highly important to preventing the most damaging strikes. The purpose of this project is to develop models to predict which strikes are likely to result in damage, injury, or death. The models utilized include decision tree, naïve Bayes, random forest (rf), support vector machines (svm), and k-means clustering algorithms. This project aims to experiment with various models and determine which one most accurately predicts which wildlife strikes results in damage to the aircraft and which results in injuries or fatalities. This project utilizes data downloaded from the Kaggle website at <https://www.kaggle.com/faa/wildlife-strikes>. The data itself was compiled from various reports that were received and published by the Federal Aviation Association.

Data Description and Preparation

The dataset contains 174,104 observations of 66 variables. The variables include an id number for each observation, the date the incident took place, the operator, what species was struck, the make and model of the aircraft and its engines, the altitude of the strike, what part of the aircraft was struck and what (if any) damage took place. Most of the data are factors, with a few numeric attributes (such as injuries, fatalities, and altitude).

Most of the wildlife strikes that were reported did not result in any damage to the aircraft (fig. 1), and most strikes were done on birds (fig. 2). Some caution is in order with regards to interpreting the accuracy of the models built, since a model that guesses that all strikes will not result in damage will result in over 90% accuracy. To properly interpret these models, it’s important to look at how well they predict damaging strikes. This point makes building a model predicting strikes resulting in injury more difficult, as only 260 of the observations resulted in injury or death.

Most of the data had to be transformed into factors, particularly the data on what part of the aircraft the strikes took place, and any damage that resulted from the strike. This data was presented as a binary “1” or “0”, but was read by R Studio as numeric. The make and model data of the engines were also presented as numeric and had to be factorized as well.

Chart, bar chart

Description automatically generatedPredicting Damaging Strikes

Fig. 1: Frequency of strikes resulting in either no damage (0) or some damage (1).

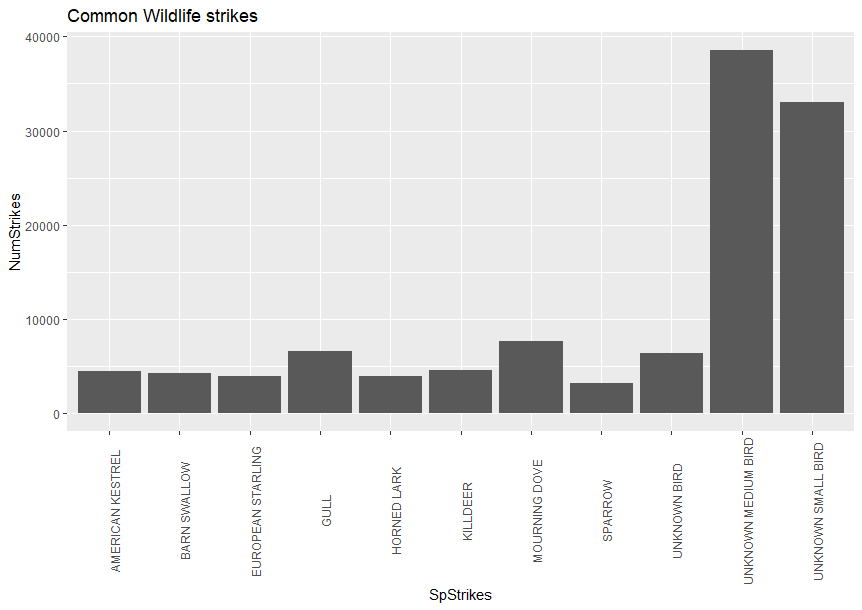
 This project attempts to use decision tree and naïve Bayes algorithms to predict which strikes resulted in damage to the aircraft, as well as support vector machines (svm) and random forest (rf). K-nearest neighbor was also attempted, however, this often resulted in the computed running out of memory. Each model was evaluated based on their accuracy in predicting damaging strikes (what the R output refers to as specificity). This is calculated by dividing the number of damaging strikes that were predicted as damaging by all the true damaging strikes.

Fig. 2: The most common species recorded being struck. Mourning Dove (*Zenaida macroura*) was the most common known species.

The ID fields for the observations were removed, so the algorithms would not simply memorize the observation IDs. The species and airport IDs were also removed, since these were redundant to the species name and airport fields, respectively. Any attributes that were likely the result of damage to the aircraft, such as injuries or deaths, were also discarded from the analysis, since these attributes would likely bias the algorithms. Finally, all the damage data to specific parts of the aircrafts were removed, since these data were redundant to the analyses.

A random sample of 25% of the observations was used in the building of the model to save on computer processing power. Of this sample, 60% were used for training the models, while 40% were used for testing them. This resulted in 26,115 observations for training and 17,411 observations for testing.

*Decision Tree and naïve Bayes*

The decision tree was produced using all the default settings, while ignoring all missing values (removing all of the missing values from the dataset would have resulted in the dataset being too small). The resulting decision tree contained 43 nodes with 13 terminal nodes. The most important variables for this model were the species name, followed by the airport, whether the animal was ingested into the engine, and the operator (table 1).

|  |  |
| --- | --- |
| Variable | Importance (%) |
| Species Name | 28 |
| Airport | 26 |
| Aircraft | 14 |
| Engine Ingested | 13 |
| Operator | 8 |
| Lights Strike | 3 |
| Aircraft Make | 2 |
| Engine2 Strike | 1 |
| Aircraft Model | 1 |
| Flight Phase | 1 |
| State | 1 |
| Engine1 Strike | 1 |

When this model was applied to the test data, it achieved 92.27% accuracy. However, it struggled to predict whether a strike would result in damage. Only 33.33% of damaging strikes were predicted as being damaging in the model (table 2).

Table 2: Confusion matrix for the decision tree model as applied to the testing data.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Observed | |
|  |  | No Damage | Damage |
| Predicted | No Damage | 15590 | 952 |
| Damage | 393 | 476 |

Table 1: Variable importance for decision tree in predicting damaging strikes.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Observed | |
|  |  | No Damage | Damage |
| Predicted | No Damage | 14484 | 708 |
| Damage | 1499 | 720 |

The naïve Bayes model, like the decision tree, was built using all of the default settings. Though the naïve Bayes model had lower accuracy than the decision tree model (87.32%), it did a better job a predicting which aircraft strikes would result in damage to the aircraft. It correctly predicted 50.42% of damaging strikes. This, however, did result in more non-damaging strikes being mistakenly categorized as damaging (Table 3).

Table 3: Confusion matrix for the naïve Bayes algorithm being tested against the test data.

*SVM and rf*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Observed | |
|  |  | No Damage | Damage |
| Predicted | No Damage | 15860 | 1299 |
| Damage | 123 | 129 |

Due to the fact that neither of these algorithms can be used on data with a lot of null values, and removing the null values would result in too few entries, these algorithms had to be used on a subset of the attributes which only included which part of the aircraft was struck. The models that resulted from these algorithms achieved high accuracies when applying them to the test data (91.83% and 91.86%, respectively). However, both of these algorithms also performed the worse when predicting the damaging strikes (9.03% and 10.64%, respectively) (Tables 4 and 5).

Table 4: Confusion matrix for the SVM algorithm when applied to the testing data.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Observed | |
|  |  | No Damage | Damage |
| Predicted | No Damage | 15842 | 1276 |
| Damage | 141 | 152 |

Table 5: Confusion matrix for the rf algorithm when applied to the testing data.

*k-means*

Various k-means algorithms were attempted with varying numbers of clusters. This was an attempt to use the clusters to identify strikes that would result in damages based on which cluster they fell into. However, for all the number of clusters attempted, none had any cluster that exclusively had damaging or non-damaging strikes (fig. 3).

Chart, bar chart

Description automatically generated

Fig. 3: k-means clusters for the highest number of centroids attempted (8). Though some clusters have more non-damaging strikes than damaging ones (e.g. 1), this might be an artifact of the few observations of damaging strikes in the dataset.

Predicting Injuries and Fatalities

Since there were even fewer strikes that resulted in injury or death (only 240 observations; hereafter referred to as harm), any algorithms attempted to predict injuries or fatalities would have a hard time detecting them. To compensate for this, the data was separated into those observations that resulted in harm and those that did not. Of those that did not result in harm, 2,160 observations were randomly selected and combined with those that did. The result was 2,400 observations that were used for constructing the models, 10% of which resulted in harm. A new attribute was then created which indicated whether a strike resulted in harm or not; producing a 1 if there was at least 1 injury or death, and 0 if there was not. The injury and fatality fields were then removed. This data was then split into training data (60%) and testing data (40%), and naïve Bayes and decision tree were applied to the data.

*Decision tree and naïve Bayes*

Both decision tree and naïve Bayes models were constructed utilizing the training data and evaluated using the testing data. The models were constructed to predict which strikes would result in any sort of harm utilizing all of the other attributes (including damages). The resulting decision tree model had the type of aircraft as the most important variable, followed by the airport and then the species name (table 6). When the decision tree model was applied to the testing data, some of the decisions resulted in a 50% split between a strike resulting in harm and no harm. These were interpreted to result in harm, since the potential price of mistakenly predicting a strike would result in no harm is much greater than mistakenly predicting that a strike would result in harm. This model resulted in 90.94% accuracy, with it being able to correctly predict 60.22% of strikes resulting in harm (table 7). When the naïve Bayes model was applied to the testing data, the model achieved 93.65% accuracy, and it correctly predicted 88.17% of strikes that resulted in harm (table 8).

|  |  |
| --- | --- |
| Variable | Importance (%) |
| Aircraft | 40 |
| Airport | 25 |
| Species Name | 16 |
| Operator | 15 |
| Aircraft Make | 2 |
| Aircraft Model | 2 |
| Flight Phase | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Observed | |
|  |  | No Harm | Harm |
| Predicted | No Harm | 817 | 37 |
| Harm | 50 | 56 |

Table 7: Confusion matrix for applying the decision tree model to predicting harm on the testing data.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Observed | |
|  |  | No Harm | Harm |
| Predicted | No Harm | 817 | 11 |
| Harm | 50 | 82 |

Table 6: Variable importance for the decision tree model in predicting harm.

Table 8: Confusion matrix for applying the naïve Bayes model to predicting harm on the testing data.

Conclusions

For predicting which strikes resulted in damages to the aircraft, the naïve Bayes model performed the best out of all models attempted. Although the decision tree model did have overall higher accuracy (92.27% vs. 87.32%), it did not perform as well at predicting strikes that resulted in damages. Since mistakenly believing that a strike would not result in damage when it in fact would has a potentially high cost, the naïve Bayes model would be preferred. The naïve Bayes model also performed the best at predicting strikes that resulted in harm, achieving both high accuracy and predicting which strikes resulted in harm (93.65% and 88.17%, respectively). The results of these experiments suggest that the naïve Bayes model is best suited for predicting the most dangerous wildlife strikes.

Appendix: code

# Needed packages

library("ggplot2")  
library(rpart)  
library(rpart.plot)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

library(rattle)

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.  
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(caret)

## Loading required package: lattice

library(caretEnsemble)

##   
## Attaching package: 'caretEnsemble'

## The following object is masked from 'package:ggplot2':  
##   
## autoplot

library(e1071)  
library(cluster)

Fist, lets load and inspect the data

setwd("C:/Users/17405/Downloads")  
StrikesFull <- read.csv("database.csv", na.strings = c("", "NA"))  
View(StrikesFull)  
dim(StrikesFull)

## [1] 174104 66

Data Munging The strikes and damage data are listed as numeric when they are binary, so they need to be changed to factors.

StrikesFull$Aircraft.Damage <- as.factor(StrikesFull$Aircraft.Damage)  
StrikesFull$Radome.Strike <- as.factor(StrikesFull$Radome.Strike)  
StrikesFull$Radome.Damage <- as.factor(StrikesFull$Radome.Damage)  
StrikesFull$Windshield.Strike <- as.factor(StrikesFull$Windshield.Strike)  
StrikesFull$Windshield.Damage <- as.factor(StrikesFull$Windshield.Damage)  
StrikesFull$Nose.Strike <- as.factor(StrikesFull$Nose.Strike)  
StrikesFull$Nose.Damage <- as.factor(StrikesFull$Nose.Damage)  
StrikesFull$Engine1.Strike <- as.factor(StrikesFull$Engine1.Strike)  
StrikesFull$Engine1.Damage <- as.factor(StrikesFull$Engine1.Damage)  
StrikesFull$Engine2.Strike <- as.factor(StrikesFull$Engine2.Strike)  
StrikesFull$Engine2.Damage <- as.factor(StrikesFull$Engine2.Damage)  
StrikesFull$Engine3.Strike <- as.factor(StrikesFull$Engine3.Strike)  
StrikesFull$Engine3.Damage <- as.factor(StrikesFull$Engine3.Damage)  
StrikesFull$Engine4.Strike <- as.factor(StrikesFull$Engine4.Strike)  
StrikesFull$Engine4.Damage <- as.factor(StrikesFull$Engine4.Damage)  
StrikesFull$Engine.Ingested <- as.factor(StrikesFull$Engine.Ingested)  
StrikesFull$Propeller.Strike <- as.factor(StrikesFull$Propeller.Strike)  
StrikesFull$Propeller.Damage <- as.factor(StrikesFull$Propeller.Damage)  
StrikesFull$Wing.or.Rotor.Strike <- as.factor(StrikesFull$Wing.or.Rotor.Strike)  
StrikesFull$Wing.or.Rotor.Damage <- as.factor(StrikesFull$Wing.or.Rotor.Damage)  
StrikesFull$Fuselage.Strike <- as.factor(StrikesFull$Fuselage.Strike)  
StrikesFull$Fuselage.Damage <- as.factor(StrikesFull$Fuselage.Damage)  
StrikesFull$Landing.Gear.Strike <- as.factor(StrikesFull$Landing.Gear.Strike)  
StrikesFull$Landing.Gear.Damage <- as.factor(StrikesFull$Landing.Gear.Damage)  
StrikesFull$Tail.Strike <- as.factor(StrikesFull$Tail.Strike)  
StrikesFull$Tail.Damage <- as.factor(StrikesFull$Tail.Damage)  
StrikesFull$Lights.Strike <- as.factor(StrikesFull$Lights.Strike)  
StrikesFull$Lights.Damage <- as.factor(StrikesFull$Lights.Damage)  
StrikesFull$Other.Strike <- as.factor(StrikesFull$Other.Strike)  
StrikesFull$Other.Damage <- as.factor(StrikesFull$Other.Damage)

Converting other attributes as needed.

StrikesFull$Engine.Make <- as.factor(StrikesFull$Engine.Make)  
StrikesFull$Engine2.Position <- as.factor(StrikesFull$Engine2.Position)  
StrikesFull$Engine4.Position <- as.factor(StrikesFull$Engine4.Position)  
StrikesFull$Species.Quantity <- as.numeric(StrikesFull$Species.Quantity)

## Warning: NAs introduced by coercion

Removing the ID field

StrikesFull <- StrikesFull[,-1]

Factorising other fields (if needed)

ColNum <- 1:ncol(StrikesFull)  
for (x in ColNum){  
 if (typeof(StrikesFull[,x]) == "character"){  
 StrikesFull[,x] <- as.factor(StrikesFull[,x])  
 }  
   
}

# There are over 174,000 observations in this dataset. We need to reduce that number, or R will run out of memory for future analyses  
percent <- .25  
set.seed(20)  
StrikesSplit <- sample(nrow(StrikesFull),nrow(StrikesFull)\*percent)  
Strikes <- StrikesFull[StrikesSplit,]  
dim(Strikes)

## [1] 43526 65

row.names(Strikes) <- NULL

Basic data exploration

# Histogram of aircraft damage.  
ggplot(StrikesFull, aes(x = Aircraft.Damage)) + geom\_bar() + ggtitle("Damaging Strikes")

Chart, bar chart

Description automatically generated

# Most of the wildlife strikes did not result in any damages.  
noDamage <- length(which(StrikesFull$Aircraft.Damage == "0"))  
Damage <- length(which(StrikesFull$Aircraft.Damage == "1"))  
(DamagePercent <- Damage/nrow(StrikesFull))

## [1] 0.08597735

(NoDamagePercent <- noDamage/nrow(StrikesFull))

## [1] 0.9140227

Only about 8.6% of wildlife strikes resuts in damages to the aircraft.

Exploring what wildlife tends to strike the aircraft.

ggplot(StrikesFull, aes(x = Species.Name)) + geom\_bar() + ggtitle("Wildlife strikes")

Histogram

Description automatically generated Observing only the most numerous species

NumStrikes <- c()  
Species <- levels(StrikesFull$Species.Name)  
for (x in Species){  
 if (length(which(StrikesFull$Species.Name == x)) >= 1000) {  
 if (length(NumStrikes) == 0){  
 NumStrikes <- c(length(which(StrikesFull$Species.Name == x)))  
 SpStrikes <- c(x)  
 }else {  
 NumStrikes <- c(NumStrikes, length(which(StrikesFull$Species.Name == x)))  
 SpStrikes <- c(SpStrikes, x)  
 }  
 }  
}  
  
SpCommonStrikes <- data.frame(SpStrikes, NumStrikes)  
  
ggplot(SpCommonStrikes, aes(x = SpStrikes, y = NumStrikes)) + geom\_col() + ggtitle("Common Wildlife strikes") +  
 theme(axis.text.x = element\_text(angle = 90))

Chart

Description automatically generated Unsurprisingly, birds are most often struck, with Mourning Dove being the most common (known) species struck.

What altitude do strikes typically occur?

summary(StrikesFull$Height)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 0 0 50 831 800 31300 70427

ggplot(StrikesFull, aes(x = Height)) + geom\_histogram(bins = 7, color = "black", fill = "white") + ggtitle("Height histogram")

## Warning: Removed 70427 rows containing non-finite values (stat\_bin).

Chart, histogram

Description automatically generated There is a strong right skew, with half of the strikes taking place at 50 (ft?) or less above the ground.

Exploring what the injuries and fatalities were like.

summary(StrikesFull$Fatalities)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 0.00 0.00 0.00 0.05 0.00 8.00 173539

summary(StrikesFull$Injuries)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 1.00 1.00 1.00 1.75 1.00 100.00 173875

# The vast majority of strikes don't have any data on injuries or fatalities   
# (hopefully because there were none).  
# What were the records with the max number of injuries and/or fatalities.  
StrikesFull[which.max(StrikesFull$Injuries), ]

## Incident.Year Incident.Month Incident.Day Operator.ID Operator Aircraft  
## 97019 2009 1 15 AWE US AIRWAYS A-320  
## Aircraft.Type Aircraft.Make Aircraft.Model Aircraft.Mass Engine.Make  
## 97019 A 04A 3 4 10  
## Engine.Model Engines Engine.Type Engine1.Position Engine2.Position  
## 97019 1 2 D 1 1  
## Engine3.Position Engine4.Position Airport.ID Airport State  
## 97019 <NA> <NA> KLGA LA GUARDIA ARPT NY  
## FAA.Region Warning.Issued Flight.Phase Visibility Precipitation Height  
## 97019 AEA <NA> CLIMB DAY NONE 2818  
## Speed Distance Species.ID Species.Name Species.Quantity Flight.Impact  
## 97019 220 4.5 J2204 CANADA GOOSE NA OTHER  
## Fatalities Injuries Aircraft.Damage Radome.Strike Radome.Damage  
## 97019 NA 100 1 0 0  
## Windshield.Strike Windshield.Damage Nose.Strike Nose.Damage  
## 97019 0 0 0 0  
## Engine1.Strike Engine1.Damage Engine2.Strike Engine2.Damage  
## 97019 1 1 1 1  
## Engine3.Strike Engine3.Damage Engine4.Strike Engine4.Damage  
## 97019 0 0 0 0  
## Engine.Ingested Propeller.Strike Propeller.Damage Wing.or.Rotor.Strike  
## 97019 1 0 0 1  
## Wing.or.Rotor.Damage Fuselage.Strike Fuselage.Damage Landing.Gear.Strike  
## 97019 0 1 0 0  
## Landing.Gear.Damage Tail.Strike Tail.Damage Lights.Strike Lights.Damage  
## 97019 0 0 0 0 0  
## Other.Strike Other.Damage  
## 97019 0 0

StrikesFull[which.max(StrikesFull$Fatalities), ]

## Incident.Year Incident.Month Incident.Day Operator.ID Operator  
## 96936 2009 1 4 PHM PHI INC  
## Aircraft Aircraft.Type Aircraft.Make Aircraft.Model Aircraft.Mass  
## 96936 SIKORSKY S-76 B 813 14 2  
## Engine.Make Engine.Model Engines Engine.Type Engine1.Position  
## 96936 43 1 2 F 6  
## Engine2.Position Engine3.Position Engine4.Position Airport.ID Airport  
## 96936 <NA> <NA> <NA> ZZZZ UNKNOWN  
## State FAA.Region Warning.Issued Flight.Phase Visibility Precipitation  
## 96936 <NA> <NA> <NA> EN ROUTE DAY <NA>  
## Height Speed Distance Species.ID Species.Name Species.Quantity  
## 96936 700 135 NA K3302 RED-TAILED HAWK 1  
## Flight.Impact Fatalities Injuries Aircraft.Damage Radome.Strike  
## 96936 <NA> 8 1 1 0  
## Radome.Damage Windshield.Strike Windshield.Damage Nose.Strike Nose.Damage  
## 96936 0 1 1 0 0  
## Engine1.Strike Engine1.Damage Engine2.Strike Engine2.Damage  
## 96936 1 0 0 0  
## Engine3.Strike Engine3.Damage Engine4.Strike Engine4.Damage  
## 96936 0 0 0 0  
## Engine.Ingested Propeller.Strike Propeller.Damage Wing.or.Rotor.Strike  
## 96936 0 0 0 0  
## Wing.or.Rotor.Damage Fuselage.Strike Fuselage.Damage Landing.Gear.Strike  
## 96936 0 0 0 0  
## Landing.Gear.Damage Tail.Strike Tail.Damage Lights.Strike Lights.Damage  
## 96936 0 0 0 0 0  
## Other.Strike Other.Damage  
## 96936 0 0

# Plot of injuries  
ggplot(StrikesFull, aes(x = Injuries)) + geom\_histogram(bins = 7, color = "black", fill = "white") + ggtitle("Injuries histogram")

## Warning: Removed 173875 rows containing non-finite values (stat\_bin).

Chart, histogram

Description automatically generated ## Question 1: What conditions of a strike is more likely to result in damage?

The first thing that needs to be done is to try and construct a model that accurately predicts what conditions of a strike are more likely to result in serious damage. We will need to be careful in constructing our algorithm, since the majority of strikes did not result in any damage, and just predicting “0” every time will give us over 90% accuracy. But first, we should separate the specific damage data from the dataset, since it’s redundant.

# Creating a backup  
Backup <- Strikes  
  
Strikes <- Strikes[,c(1:50, 52:65, 51)] # Reordering the columns so that Engine.Ingested is the last column (ensures strike columns are odd while damage columns are even)  
StrikeCond <- Strikes[,1:36]  
StrikeDamage <- data.frame(c(1:which.max(row.names(Strikes))))  
I <- c(37:65) # Columns that need to be checked.  
Damage <- c()  
Condition <- c()  
Init <- colnames(StrikeCond)  
# The damage columns need to be removed, while keeping the strike columns. Thankfully, the former column numbers are even, while the latter are odd.  
for (x in I){  
 if (x %% 2 == 0){  
 StrikeDamage <- data.frame(StrikeDamage, Strikes[,x])  
 Damage <- c(Damage, colnames(Strikes[x]))  
 }else{  
 StrikeCond <- data.frame(StrikeCond, Strikes[,x])  
 Condition <- c(Condition, colnames(Strikes[x]))  
 }  
}  
StrikeDamage <- StrikeDamage[,-1]  
colnames(StrikeDamage) <- Damage  
colnames(StrikeCond) <- c(Init, Condition)

One final look at the structure of the data before we make any changes.

str(StrikeCond)

## 'data.frame': 43526 obs. of 51 variables:  
## $ Incident.Year : int 2010 2015 2012 2005 2008 2002 2014 1992 2011 2005 ...  
## $ Incident.Month : int 11 4 5 8 3 9 10 10 5 5 ...  
## $ Incident.Day : int 8 12 19 30 28 5 1 23 22 18 ...  
## $ Operator.ID : Factor w/ 539 levels "1AAH","1ASQ",..: 210 484 498 184 184 149 356 295 445 493 ...  
## $ Operator : Factor w/ 533 levels "1US AIRWAYS",..: 245 165 499 224 224 182 374 334 448 494 ...  
## $ Aircraft : Factor w/ 656 levels "A-10","A-10A",..: 282 317 22 3 3 78 22 451 69 637 ...  
## $ Aircraft.Type : Factor w/ 3 levels "A","B","J": 1 1 2 1 1 1 2 1 1 NA ...  
## $ Aircraft.Make : Factor w/ 95 levels "04a","04A","100",..: 16 27 28 2 2 11 28 11 11 NA ...  
## $ Aircraft.Model : Factor w/ 63 levels "0","1","10","11",..: 9 3 8 2 2 20 8 NA 38 NA ...  
## $ Aircraft.Mass : int 4 3 2 4 4 4 2 NA 4 NA ...  
## $ Engine.Make : Factor w/ 34 levels "1","2","3","7",..: 15 21 29 24 24 24 29 NA 6 NA ...  
## $ Engine.Model : Factor w/ 64 levels "??","0","1","1-",..: 28 5 3 NA 33 29 3 NA 3 NA ...  
## $ Engines : int 2 2 1 2 2 2 1 NA 2 NA ...  
## $ Engine.Type : Factor w/ 9 levels "A","A/C","B",..: 7 6 8 7 7 7 8 NA 7 NA ...  
## $ Engine1.Position : Factor w/ 8 levels "1","2","3","4",..: 5 4 NA 1 1 1 NA NA 1 NA ...  
## $ Engine2.Position : Factor w/ 7 levels "1","2","3","4",..: 5 4 NA 1 1 1 NA NA 1 NA ...  
## $ Engine3.Position : Factor w/ 5 levels "1","3","4","5",..: NA NA NA NA NA NA NA NA NA NA ...  
## $ Engine4.Position : Factor w/ 4 levels "1","3","4","5": NA NA NA NA NA NA NA NA NA NA ...  
## $ Airport.ID : Factor w/ 2228 levels "00C","00M","00OI",..: 741 1044 2228 1220 1631 521 2228 1221 484 996 ...  
## $ Airport : Factor w/ 2226 levels "ABERDEEN REGIONAL AR",..: 479 2123 2066 1261 1070 812 2066 281 24 787 ...  
## $ State : Factor w/ 62 levels "AB","AK","AL",..: 8 10 NA 53 28 13 NA 7 40 24 ...  
## $ FAA.Region : Factor w/ 15 levels "AAL","ACE","AEA",..: 6 3 NA 7 2 7 NA 10 3 3 ...  
## $ Warning.Issued : Factor w/ 4 levels "n","N","y","Y": NA 2 2 NA NA 4 2 NA 2 NA ...  
## $ Flight.Phase : Factor w/ 12 levels "APPROACH","ARRIVAL",..: 8 1 6 1 1 1 6 11 1 NA ...  
## $ Visibility : Factor w/ 5 levels "DAWN","DAY","DUSK",..: 4 NA 2 4 4 2 4 2 4 NA ...  
## $ Precipitation : Factor w/ 8 levels "FOG","FOG, RAIN",..: NA 5 5 NA NA 5 5 NA 5 NA ...  
## $ Height : int 0 4000 200 500 350 4500 500 0 2200 NA ...  
## $ Speed : int NA 210 80 NA NA 210 100 160 210 NA ...  
## $ Distance : num 0 5 NA NA NA NA NA 0 7 0 ...  
## $ Species.ID : Factor w/ 719 levels "100000000000",..: 36 449 226 452 451 453 453 452 452 496 ...  
## $ Species.Name : Factor w/ 715 levels "ACADIAN FLYCATCHER",..: 64 620 307 623 622 624 624 623 623 41 ...  
## $ Species.Quantity : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ Flight.Impact : Factor w/ 6 levels "ABORTED TAKEOFF",..: NA 4 4 4 4 4 4 NA 5 NA ...  
## $ Fatalities : int NA NA NA NA NA NA NA NA NA NA ...  
## $ Injuries : int NA NA NA NA NA NA NA NA NA NA ...  
## $ Aircraft.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...  
## $ Radome.Strike : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Windshield.Strike : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 2 2 1 ...  
## $ Nose.Strike : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 2 1 1 1 ...  
## $ Engine1.Strike : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Engine2.Strike : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Engine3.Strike : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Engine4.Strike : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Propeller.Strike : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Wing.or.Rotor.Strike: Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 1 1 1 ...  
## $ Fuselage.Strike : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Landing.Gear.Strike : Factor w/ 2 levels "0","1": 2 1 1 1 2 1 1 1 1 1 ...  
## $ Tail.Strike : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Lights.Strike : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...  
## $ Other.Strike : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Engine.Ingested : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

str(StrikeDamage)

## 'data.frame': 43526 obs. of 14 variables:  
## $ Radome.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Windshield.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Nose.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Engine1.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Engine2.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Engine3.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Engine4.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Propeller.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Wing.or.Rotor.Damage: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Fuselage.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Landing.Gear.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Tail.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Lights.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...  
## $ Other.Damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

Since the main purpose of this project is to try and predict which attributes of a strike are more likely to result in damage, out models should disregard any fields that are likely the results of the damage, such as any injuries ore fatalities.

StrikeCond <- StrikeCond[,-33]  
StrikeCond <- StrikeCond[,-33]  
StrikeCond <- StrikeCond[,-33]  
  
# Species ID should also be removed, since we have the species name. The warning.issued column is also more likely a result of damage.  
StrikeCond <- StrikeCond[,-30]  
StrikeCond <- StrikeCond[,-23]  
  
# Year and day of incident will be removed, since these attributes will not yield any interesting insights. Month, however, will stay, since seasonal differences might make a difference.  
StrikeCond <- StrikeCond[,-1]  
StrikeCond <- StrikeCond[,-2]  
  
# Airport and operator ID are likely to result in the same problems as species ID  
StrikeCond <- StrikeCond[,-2]  
StrikeCond <- StrikeCond[,-16]

Descretizing the height variable will help improve the models. While deciding how to break up the data will be tricky, since there is a strong right-skew to the data.

summary(StrikeCond$Height)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 0.0 0.0 50.0 829.7 800.0 31300.0 17606

# However, we will need the height variable to be continuous for the clustering algorithm, so we should create a backup with the continuous data.  
Backup2 <- StrikeCond  
  
# Bins for descritization: 0-99; 100-999; 1000-5000; 5000-9999; >= 10000  
StrikeCond$Height <- cut(StrikeCond$Height, breaks = c(0, 100, 1000, 5000, 10000, Inf), labels = c("0-99", "100-999", "1000-4999", "5000-9999", ">=10000"))  
  
str(StrikeCond$Height)

## Factor w/ 5 levels "0-99","100-999",..: NA 3 2 2 2 3 2 NA 3 NA ...

# Looking at what the descritized height data looks like.  
ggplot(data = subset(StrikeCond, !is.na(Height)), aes(x = Height)) + geom\_bar() + ggtitle("Height of Strikes")

Chart, bar chart

Description automatically generated Now the data is ready to start building some models. The first model that will be built is a decision tree.

## Decision Tree

The first model that will be evaluated will be the decision tree. Since there are over 174,000 observations in this dataset, we will use a holdout test.

trainRatio <- .60  
set.seed(20)   
sample <- sample.int(n = nrow(StrikeCond), size = floor(trainRatio\*nrow(StrikeCond)), replace = FALSE)  
train <- StrikeCond[sample, ]  
test <- StrikeCond[-sample, ]  
# train / test ratio  
length(sample)/nrow(StrikeCond)

## [1] 0.5999862

Creating the decision tree

set.seed(20)  
tree.cond <- rpart(Aircraft.Damage ~ ., data = train, method = "class", na.action = na.pass)  
summary(tree.cond)

## Call:  
## rpart(formula = Aircraft.Damage ~ ., data = train, na.action = na.pass,   
## method = "class")  
## n= 26115   
##   
## CP nsplit rel error xerror xstd  
## 1 0.08801903 0 1.0000000 1.0000000 0.01985532  
## 2 0.03070934 2 0.8239619 0.9204152 0.01912234  
## 3 0.02984429 4 0.7625433 0.9178201 0.01909775  
## 4 0.02032872 5 0.7326990 0.9117647 0.01904020  
## 5 0.01859862 7 0.6920415 0.9113322 0.01903608  
## 6 0.01297578 9 0.6548443 0.9173875 0.01909364  
## 7 0.01254325 10 0.6418685 0.9204152 0.01912234  
## 8 0.01038062 11 0.6293253 0.9052768 0.01897826  
## 9 0.01000000 12 0.6189446 0.9022491 0.01894926  
##   
## Variable importance  
## Species.Name Airport Aircraft Engine.Ingested Operator   
## 28 26 14 13 8   
## Lights.Strike Aircraft.Make Engine2.Strike Aircraft.Model Flight.Phase   
## 3 2 1 1 1   
## State Engine1.Strike   
## 1 1   
##   
## Node number 1: 26115 observations, complexity param=0.08801903  
## predicted class=0 expected loss=0.0885315 P(node) =1  
## class counts: 23803 2312  
## probabilities: 0.911 0.089   
## left son=2 (24604 obs) right son=3 (1511 obs)  
## Primary splits:  
## Species.Name, improve=581.1556, (15 missing)  
## Airport, improve=455.2340, (38 missing)  
## Engine.Ingested, improve=396.7990, (0 missing)  
## Aircraft, improve=379.6092, (0 missing)  
## Operator, improve=320.9689, (0 missing)  
## Surrogate splits:  
## Airport, agree=0.949, adj=0.116, (13 split)  
## Aircraft, agree=0.943, adj=0.023, (2 split)  
## Operator, agree=0.943, adj=0.012, (0 split)  
##   
## Node number 2: 24604 observations, complexity param=0.03070934  
## predicted class=0 expected loss=0.06238823 P(node) =0.9421405  
## class counts: 23069 1535  
## probabilities: 0.938 0.062   
## left son=4 (23351 obs) right son=5 (1253 obs)  
## Primary splits:  
## Engine.Ingested, improve=316.5289, (0 missing)  
## Airport, improve=169.7195, (37 missing)  
## Aircraft, improve=158.4617, (0 missing)  
## Species.Name, improve=130.3689, (15 missing)  
## Engine1.Strike, improve=118.8243, (0 missing)  
## Surrogate splits:  
## Engine2.Strike, agree=0.954, adj=0.101, (0 split)  
## Engine1.Strike, agree=0.952, adj=0.052, (0 split)  
## Operator, agree=0.950, adj=0.014, (0 split)  
## Aircraft, agree=0.949, adj=0.006, (0 split)  
## Engine3.Strike, agree=0.949, adj=0.002, (0 split)  
##   
## Node number 3: 1511 observations, complexity param=0.08801903  
## predicted class=1 expected loss=0.485771 P(node) =0.05785947  
## class counts: 734 777  
## probabilities: 0.486 0.514   
## left son=6 (984 obs) right son=7 (527 obs)  
## Primary splits:  
## Airport, improve=224.91990, (1 missing)  
## Aircraft, improve=155.06590, (0 missing)  
## Operator, improve=133.43350, (0 missing)  
## Aircraft.Make, improve= 77.71705, (130 missing)  
## Species.Name, improve= 64.03948, (0 missing)  
## Surrogate splits:  
## Aircraft, agree=0.803, adj=0.433, (1 split)  
## Operator, agree=0.781, adj=0.371, (0 split)  
## Species.Name, agree=0.729, adj=0.222, (0 split)  
## Aircraft.Make, agree=0.691, adj=0.112, (0 split)  
## Engine.Type, agree=0.670, adj=0.051, (0 split)  
##   
## Node number 4: 23351 observations, complexity param=0.02984429  
## predicted class=0 expected loss=0.04380969 P(node) =0.8941604  
## class counts: 22328 1023  
## probabilities: 0.956 0.044   
## left son=8 (23172 obs) right son=9 (179 obs)  
## Primary splits:  
## Airport, improve=151.91280, (34 missing)  
## Aircraft, improve=117.50410, (0 missing)  
## Lights.Strike splits as LR, improve= 94.32415, (0 missing)  
## Operator, improve= 89.98863, (0 missing)  
## Aircraft.Make, improve= 69.61337, (6282 missing)  
## Surrogate splits:  
## Aircraft, agree=0.993, adj=0.034, (34 split)  
## Operator, agree=0.993, adj=0.028, (0 split)  
##   
## Node number 5: 1253 observations, complexity param=0.03070934  
## predicted class=0 expected loss=0.4086193 P(node) =0.04798009  
## class counts: 741 512  
## probabilities: 0.591 0.409   
## left son=10 (1041 obs) right son=11 (212 obs)  
## Primary splits:  
## Aircraft, improve=92.74042, (0 missing)  
## Airport, improve=81.80180, (3 missing)  
## Flight.Phase splits as L-RRRR-LLLRL, improve=60.60486, (358 missing)  
## Species.Name, improve=60.28670, (0 missing)  
## Operator, improve=57.70194, (0 missing)  
## Surrogate splits:  
## Airport, agree=0.875, adj=0.259, (0 split)  
## Operator, agree=0.855, adj=0.142, (0 split)  
## Aircraft.Make, agree=0.847, adj=0.094, (0 split)  
## Species.Name, agree=0.837, adj=0.038, (0 split)  
##   
## Node number 6: 984 observations, complexity param=0.01859862  
## predicted class=0 expected loss=0.3150407 P(node) =0.03767949  
## class counts: 674 310  
## probabilities: 0.685 0.315   
## left son=12 (455 obs) right son=13 (529 obs)  
## Primary splits:  
## Aircraft, improve=59.55235, (0 missing)  
## Engine.Ingested, improve=52.88680, (0 missing)  
## Airport, improve=45.79612, (0 missing)  
## Operator, improve=39.85654, (0 missing)  
## Wing.or.Rotor.Strike, improve=35.39824, (0 missing)  
## Surrogate splits:  
## Operator, agree=0.740, adj=0.437, (0 split)  
## Aircraft.Model, agree=0.735, adj=0.426, (0 split)  
## Airport, agree=0.699, adj=0.349, (0 split)  
## Species.Name, agree=0.592, adj=0.119, (0 split)  
## Aircraft.Make, agree=0.590, adj=0.114, (0 split)  
##   
## Node number 7: 527 observations  
## predicted class=1 expected loss=0.113852 P(node) =0.02017997  
## class counts: 60 467  
## probabilities: 0.114 0.886   
##   
## Node number 8: 23172 observations, complexity param=0.01254325  
## predicted class=0 expected loss=0.03879682 P(node) =0.8873061  
## class counts: 22273 899  
## probabilities: 0.961 0.039   
## left son=16 (23061 obs) right son=17 (111 obs)  
## Primary splits:  
## Lights.Strike, improve=78.13361, (0 missing)  
## Aircraft, improve=76.93836, (0 missing)  
## Airport, improve=54.83514, (34 missing)  
## Species.Name, improve=54.61194, (15 missing)  
## Operator, improve=53.78643, (0 missing)  
## Surrogate splits:  
## Operator, agree=0.995, adj=0.009, (0 split)  
##   
## Node number 9: 179 observations, complexity param=0.01297578  
## predicted class=1 expected loss=0.3072626 P(node) =0.006854298  
## class counts: 55 124  
## probabilities: 0.307 0.693   
## left son=18 (62 obs) right son=19 (117 obs)  
## Primary splits:  
## Aircraft, improve=35.84380, (0 missing)  
## Airport, improve=19.74094, (0 missing)  
## Species.Name, improve=13.45413, (0 missing)  
## Operator, improve=12.63748, (0 missing)  
## Aircraft.Model, improve= 8.60517, (28 missing)  
## Surrogate splits:  
## Airport, agree=0.771, adj=0.339, (0 split)  
## Operator, agree=0.754, adj=0.290, (0 split)  
## Species.Name, agree=0.737, adj=0.242, (0 split)  
## Aircraft.Make, agree=0.721, adj=0.194, (0 split)  
## Aircraft.Model, agree=0.715, adj=0.177, (0 split)  
##   
## Node number 10: 1041 observations, complexity param=0.02032872  
## predicted class=0 expected loss=0.321806 P(node) =0.03986215  
## class counts: 706 335  
## probabilities: 0.678 0.322   
## left son=20 (679 obs) right son=21 (362 obs)  
## Primary splits:  
## Airport, improve=60.64090, (3 missing)  
## Flight.Phase, improve=43.82031, (328 missing)  
## Species.Name, improve=40.51767, (0 missing)  
## Aircraft, improve=37.59058, (0 missing)  
## Operator, improve=32.79549, (0 missing)  
## Surrogate splits:  
## Species.Name, agree=0.718, adj=0.188, (2 split)  
## Operator, agree=0.697, adj=0.127, (1 split)  
## Aircraft, agree=0.678, adj=0.075, (0 split)  
## Radome.Strike, agree=0.657, adj=0.014, (0 split)  
## Lights.Strike, agree=0.655, adj=0.008, (0 split)  
##   
## Node number 11: 212 observations  
## predicted class=1 expected loss=0.1650943 P(node) =0.00811794  
## class counts: 35 177  
## probabilities: 0.165 0.835   
##   
## Node number 12: 455 observations  
## predicted class=0 expected loss=0.1274725 P(node) =0.01742294  
## class counts: 397 58  
## probabilities: 0.873 0.127   
##   
## Node number 13: 529 observations, complexity param=0.01859862  
## predicted class=0 expected loss=0.4763705 P(node) =0.02025656  
## class counts: 277 252  
## probabilities: 0.524 0.476   
## left son=26 (207 obs) right son=27 (322 obs)  
## Primary splits:  
## Airport, improve=40.65460, (0 missing)  
## Wing.or.Rotor.Strike, improve=27.50171, (0 missing)  
## Engine.Ingested, improve=25.97021, (0 missing)  
## Operator, improve=22.54816, (0 missing)  
## Aircraft, improve=21.81091, (0 missing)  
## Surrogate splits:  
## State, agree=0.813, adj=0.522, (0 split)  
## Operator, agree=0.720, adj=0.285, (0 split)  
## FAA.Region, agree=0.665, adj=0.145, (0 split)  
## Aircraft, agree=0.650, adj=0.106, (0 split)  
## Species.Name, agree=0.622, adj=0.034, (0 split)  
##   
## Node number 16: 23061 observations  
## predicted class=0 expected loss=0.03594814 P(node) =0.8830557  
## class counts: 22232 829  
## probabilities: 0.964 0.036   
##   
## Node number 17: 111 observations, complexity param=0.01038062  
## predicted class=1 expected loss=0.3693694 P(node) =0.004250431  
## class counts: 41 70  
## probabilities: 0.369 0.631   
## left son=34 (24 obs) right son=35 (87 obs)  
## Primary splits:  
## Airport, improve=24.15729, (1 missing)  
## Aircraft, improve=18.34390, (0 missing)  
## Operator, improve=14.41441, (0 missing)  
## Species.Name, improve=10.70161, (0 missing)  
## State, improve=10.33910, (33 missing)  
## Surrogate splits:  
## Aircraft, agree=0.900, adj=0.542, (1 split)  
## Operator, agree=0.891, adj=0.500, (0 split)  
## Aircraft.Make, agree=0.836, adj=0.250, (0 split)  
## Species.Name, agree=0.827, adj=0.208, (0 split)  
## Radome.Strike, agree=0.800, adj=0.083, (0 split)  
##   
## Node number 18: 62 observations  
## predicted class=0 expected loss=0.2580645 P(node) =0.002374114  
## class counts: 46 16  
## probabilities: 0.742 0.258   
##   
## Node number 19: 117 observations  
## predicted class=1 expected loss=0.07692308 P(node) =0.004480184  
## class counts: 9 108  
## probabilities: 0.077 0.923   
##   
## Node number 20: 679 observations  
## predicted class=0 expected loss=0.197349 P(node) =0.02600038  
## class counts: 545 134  
## probabilities: 0.803 0.197   
##   
## Node number 21: 362 observations, complexity param=0.02032872  
## predicted class=1 expected loss=0.4447514 P(node) =0.01386177  
## class counts: 161 201  
## probabilities: 0.445 0.555   
## left son=42 (148 obs) right son=43 (214 obs)  
## Primary splits:  
## Flight.Phase, improve=30.13902, (17 missing)  
## Aircraft, improve=24.88751, (0 missing)  
## Species.Name, improve=23.56377, (0 missing)  
## Operator, improve=22.22146, (0 missing)  
## Airport, improve=15.06464, (1 missing)  
## Surrogate splits:  
## Airport, agree=0.704, adj=0.292, (16 split)  
## Aircraft, agree=0.684, adj=0.243, (1 split)  
## Species.Name, agree=0.678, adj=0.229, (0 split)  
## Operator, agree=0.667, adj=0.201, (0 split)  
## FAA.Region, agree=0.623, adj=0.097, (0 split)  
##   
## Node number 26: 207 observations  
## predicted class=0 expected loss=0.2318841 P(node) =0.007926479  
## class counts: 159 48  
## probabilities: 0.768 0.232   
##   
## Node number 27: 322 observations  
## predicted class=1 expected loss=0.3664596 P(node) =0.01233008  
## class counts: 118 204  
## probabilities: 0.366 0.634   
##   
## Node number 34: 24 observations  
## predicted class=0 expected loss=0 P(node) =0.0009190121  
## class counts: 24 0  
## probabilities: 1.000 0.000   
##   
## Node number 35: 87 observations  
## predicted class=1 expected loss=0.1954023 P(node) =0.003331419  
## class counts: 17 70  
## probabilities: 0.195 0.805   
##   
## Node number 42: 148 observations  
## predicted class=0 expected loss=0.3175676 P(node) =0.005667241  
## class counts: 101 47  
## probabilities: 0.682 0.318   
##   
## Node number 43: 214 observations  
## predicted class=1 expected loss=0.2803738 P(node) =0.008194524  
## class counts: 60 154  
## probabilities: 0.280 0.720

fancyRpartPlot(tree.cond)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting

A picture containing text, sky, screenshot, document

Description automatically generated The decision tree identified the species name as the most important attribute for predicting whether a strike will result in damage. Applying the decision tree to the test data.

tree.damage.pred <- data.frame(predict(tree.cond, test))  
  
# Since the prediction assigns probabilities that there was damage or not, the predicted value (0 or 1) needs to be assigned.  
K <- c(1:nrow(tree.damage.pred))  
P <- c()  
for (x in K){  
 if(tree.damage.pred[x,1] > 0.5){  
 P <- c(P, 0)}  
 else {  
 P <- c(P, 1)  
 }  
}  
  
tree.damage.pred$pred <- as.factor(P)  
  
# Confusion matrix  
(tree.damage.conf <- confusionMatrix(tree.damage.pred$pred, test$Aircraft.Damage))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 15590 952  
## 1 393 476  
##   
## Accuracy : 0.9227   
## 95% CI : (0.9187, 0.9267)  
## No Information Rate : 0.918   
## P-Value [Acc > NIR] : 0.01086   
##   
## Kappa : 0.3757   
##   
## Mcnemar's Test P-Value : < 2e-16   
##   
## Sensitivity : 0.9754   
## Specificity : 0.3333   
## Pos Pred Value : 0.9424   
## Neg Pred Value : 0.5478   
## Prevalence : 0.9180   
## Detection Rate : 0.8954   
## Detection Prevalence : 0.9501   
## Balanced Accuracy : 0.6544   
##   
## 'Positive' Class : 0   
##

Unfortunately, it looks like this model is struggling to detect when a strike is likely to result in damage (only 33% of damaging strikes were detected). The model might be too generic, and a more complicated model will likely need to be constructed.

# Naive Bayes

set.seed(20)  
nb.cond <- naiveBayes(Aircraft.Damage ~., data = train, laplace = 1, na.action = na.pass)  
summary(nb.cond)

## Length Class Mode   
## apriori 2 table numeric   
## tables 41 -none- list   
## levels 2 -none- character  
## isnumeric 41 -none- logical   
## call 4 -none- call

# Naive Bayes prediction

nb.damage.pred <- data.frame(predict(nb.cond, test))  
(nb.damage.conf <- confusionMatrix(nb.damage.pred$predict.nb.cond..test., test$Aircraft.Damage))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 14550 739  
## 1 1433 689  
##   
## Accuracy : 0.8753   
## 95% CI : (0.8703, 0.8801)  
## No Information Rate : 0.918   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3217   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9103   
## Specificity : 0.4825   
## Pos Pred Value : 0.9517   
## Neg Pred Value : 0.3247   
## Prevalence : 0.9180   
## Detection Rate : 0.8357   
## Detection Prevalence : 0.8781   
## Balanced Accuracy : 0.6964   
##   
## 'Positive' Class : 0   
##

The naive Bayes model fared better at predicting damaging strikes, correctly identifying 50% of the damaging strikes.

# SVM, knn, and random forest

Unfortunately, since this dataset has so many NA’s, SVM, knn, and rf will be difficult to implement with the entire dataset. However, they may still be used by modeling the strike data alone, and determine whether certain parts of the plane that are struck are more likely to cause damage to the aircraft. Even with this, knn did not work with this dataset, so only svm and random forest were analyzed.

defaultW <- getOption("warn")   
options(warn = -1)   
  
# Preparing data  
#Ptrain <- train[,27:42]  
#Ptest <- test[,27:42]  
  
#set.seed(20)  
#SVM.cond <- train(Aircraft.Damage ~ ., data = Ptrain, method = "svmRadial", na.action = na.omit)  
#set.seed(20)  
#rf.cond <- train(Aircraft.Damage ~ ., data = Ptrain, method = "rf", na.action = na.omit)  
  
# WARNING: these models took a very long time to produce (around an hour). They have been commented out to save time knitting this document.  
options(warn = defaultW)

# Model predictions

#SVM.damage.pred <- data.frame(predict(SVM.cond, test))  
#rf.damage.pred <- data.frame(predict(rf.cond, test))  
  
#(SVM.damage.conf <- confusionMatrix(SVM.damage.pred$predict.SVM.cond..test., test$Aircraft.Damage))  
#(rf.damage.conf <- confusionMatrix(rf.damage.pred$predict.rf.cond..test., test$Aircraft.Damage))  
  
# See above warning.

Neither svm nor random forest performed well in terms of predicting damaging strikes, only predicting 11% and 9% of damaging strikes, respectively. These analyses might not be appropriate for this particular for this particular question.

## K-means clustering

Now to perform a clustering algorithm to see if damaging strikes fall into seperate clusters.

# Returning our height variable to continuous data  
sum(is.na(StrikeCond))

## [1] 344286

# Unfortunately, there are a lot of NA's in this dataset, which kmeans can't handle, and removing them from the dataset will likely make us lose too much data. However, using only the data of which part of the plane was struck, we can avoid this problem.  
PStrikeCond <- StrikeCond[,27:42]  
sum(is.na(PStrikeCond))

## [1] 0

# Run k-means  
set.seed(20)  
Clusters <- kmeans(PStrikeCond[,2:16], 8) # Don't want to include the attribute we are trying to predict in the algorithm.  
PStrikeCond$cluster <- as.factor(Clusters$cluster)   
ggplot(data = PStrikeCond, aes(x = Aircraft.Damage, fill = cluster)) + geom\_bar(stat = "count") + labs(title = "Damage")

Chart, bar chart

Description automatically generated There does not seem to be any cluster that can accurately identify whether a strike will result in damage or not, and there does not seem to be any particular part of the aircraft that would be more likely to cause damage to the aircraft than any other.

## Strikes Resulting in Injury

The next task will be to try and predict which strikes are the most dangerous to those in the aircraft. The same methods as before will be utilized, but instead predicting which strikes result in injuries or deaths on the aircraft.

# Data preperation

length(which(StrikesFull$Injuries > 0 | StrikesFull$Fatalities > 0))/nrow(StrikesFull)

## [1] 0.001378486

# Since very few strikes result in injuries or death, it might be prudent to trim the data a bit by removing those strikes that did not result in any damage, since they are unlikely to result in any injuries.  
  
Damage <- which(StrikesFull$Aircraft.Damage == "1")  
StrikesD <- StrikesFull[Damage,]  
  
# Did this result in us losing any strikes resulting in injury or death?  
length(which(StrikesFull$Injuries > 0 | StrikesFull$Fatalities > 0))

## [1] 240

length(which(StrikesD$Injuries > 0 | StrikesD$Fatalities > 0))

## [1] 234

# Unfortunately, most of the strikes that resulted in injury or death were removed using this method; proving the initial assumption wrong.  
  
# Separating observations with injuries or fatalities from those that don't  
Injuries <- which(StrikesFull$Injuries > 0 | StrikesFull$Fatalities > 0)  
  
# Since the goal is to predict whether a strike will result in any physical harm, a new column needs to be created to act as the predictor variable that isn't numeric.  
  
# Replacing NA's with 0's; assuming that any NA's means that there were no injuries or deaths.  
StrikesFRows <- c(1:nrow(StrikesFull))  
  
Backup3 <- StrikesFull  
  
StrikesFull[["Injuries"]][is.na(StrikesFull[["Injuries"]])] <- 0  
StrikesFull[["Fatalities"]][is.na(StrikesFull[["Fatalities"]])] <- 0  
  
# Making a separate column identifying whether a strike resulted in injury or fatality  
  
IorD <- c()  
for (x in StrikesFRows){  
 if (StrikesFull[x,]$Injuries > 0 | StrikesFull[x,]$Fatalities > 0 ){  
 IorD <- c(IorD, "1")  
 }  
 else{  
 IorD <- c(IorD, "0")  
 }  
}  
  
StrikesFull$IorD <- IorD  
length(which(StrikesFull$IorD == "1"))

## [1] 240

StrikesFull$IorD <- as.factor(StrikesFull$IorD)

Since the data is heavily skewed towards flights not resulting in injury, the data should be sampled to improve the ratio.

# Separating strikes with injuries from those that don't.  
StrikesI <- StrikesFull[which(StrikesFull$IorD == "1"),]  
StrikesN <- StrikesFull[which(StrikesFull$IorD == "0"),]  
  
# Selecting 2160 of non-damaging strikes, so 10% of strikes are damaging.  
set.seed(20)  
NSample <- sample(nrow(StrikesN), 2160, replace = FALSE)  
StrikesN <- StrikesN[NSample,]  
  
# Rejoining the two data frames.  
StrikesIorD <- rbind(StrikesI, StrikesN)  
# Removing redundant columns  
StrikesIorD <- StrikesIorD[,-34]  
StrikesIorD <- StrikesIorD[,-34]  
  
# Removing redundant data  
StrikesIorD <- StrikesIorD[,-1]  
StrikesIorD <- StrikesIorD[,-2]  
StrikesIorD <- StrikesIorD[,-2]  
StrikesIorD <- StrikesIorD[,-16]  
StrikesIorD <- StrikesIorD[,-26]

## Injuring strikes models

# Setting up training and test data  
trainRatio <- .60  
set.seed(20)   
sample2 <- sample.int(n = nrow(StrikesIorD), size = floor(trainRatio\*nrow(StrikesIorD)), replace = FALSE)  
trainI <- StrikesIorD[sample2, ]  
testI <- StrikesIorD[-sample2, ]  
# train / test ratio  
length(sample2)/nrow(StrikesIorD)

## [1] 0.6

# Decision Tree  
set.seed(20)  
tree.inj <- rpart(IorD ~ ., data = trainI, method = "class", na.action = na.pass)  
summary(tree.inj)

## Call:  
## rpart(formula = IorD ~ ., data = trainI, na.action = na.pass,   
## method = "class")  
## n= 1440   
##   
## CP nsplit rel error xerror xstd  
## 1 0.76190476 0 1.0000000 1.0000000 0.07815546  
## 2 0.10204082 1 0.2380952 0.2380952 0.03975334  
## 3 0.01360544 2 0.1360544 0.2925170 0.04393735  
## 4 0.01000000 3 0.1224490 0.2993197 0.04442940  
##   
## Variable importance  
## Windshield.Damage Aircraft Species.Name Airport   
## 39 27 13 12   
## Operator Nose.Damage Fuselage.Damage   
## 4 4 1   
##   
## Node number 1: 1440 observations, complexity param=0.7619048  
## predicted class=0 expected loss=0.1020833 P(node) =1  
## class counts: 1293 147  
## probabilities: 0.898 0.102   
## left son=2 (1322 obs) right son=3 (118 obs)  
## Primary splits:  
## Windshield.Damage, improve=195.6892, (0 missing)  
## Aircraft, improve=151.1266, (0 missing)  
## Aircraft.Damage, improve=131.3415, (0 missing)  
## Flight.Impact, improve=113.5203, (584 missing)  
## Operator, improve=105.8679, (0 missing)  
## Surrogate splits:  
## Aircraft, agree=0.963, adj=0.542, (0 split)  
## Species.Name, agree=0.944, adj=0.322, (0 split)  
## Airport, agree=0.937, adj=0.229, (0 split)  
## Operator, agree=0.926, adj=0.102, (0 split)  
## Nose.Damage, agree=0.926, adj=0.093, (0 split)  
##   
## Node number 2: 1322 observations, complexity param=0.1020408  
## predicted class=0 expected loss=0.02420575 P(node) =0.9180556  
## class counts: 1290 32  
## probabilities: 0.976 0.024   
## left son=4 (1303 obs) right son=5 (19 obs)  
## Primary splits:  
## Aircraft, improve=29.21724, (0 missing)  
## Airport, improve=27.28530, (3 missing)  
## Flight.Impact, improve=20.55275, (572 missing)  
## Species.Name, improve=18.06580, (0 missing)  
## Fuselage.Damage, improve=17.06339, (0 missing)  
## Surrogate splits:  
## Airport, agree=0.992, adj=0.474, (0 split)  
## Species.Name, agree=0.987, adj=0.105, (0 split)  
##   
## Node number 3: 118 observations  
## predicted class=1 expected loss=0.02542373 P(node) =0.08194444  
## class counts: 3 115  
## probabilities: 0.025 0.975   
##   
## Node number 4: 1303 observations, complexity param=0.01360544  
## predicted class=0 expected loss=0.0115119 P(node) =0.9048611  
## class counts: 1288 15  
## probabilities: 0.988 0.012   
## left son=8 (1293 obs) right son=9 (10 obs)  
## Primary splits:  
## Fuselage.Damage, improve=6.979933, (0 missing)  
## Airport, improve=6.841400, (3 missing)  
## Species.Name, improve=6.467809, (0 missing)  
## Flight.Impact, improve=5.583641, (571 missing)  
## Propeller.Damage, improve=4.354896, (0 missing)  
## Surrogate splits:  
## Aircraft, agree=0.995, adj=0.3, (0 split)  
## Operator, agree=0.994, adj=0.2, (0 split)  
## Species.Name, agree=0.993, adj=0.1, (0 split)  
##   
## Node number 5: 19 observations  
## predicted class=1 expected loss=0.1052632 P(node) =0.01319444  
## class counts: 2 17  
## probabilities: 0.105 0.895   
##   
## Node number 8: 1293 observations  
## predicted class=0 expected loss=0.006960557 P(node) =0.8979167  
## class counts: 1284 9  
## probabilities: 0.993 0.007   
##   
## Node number 9: 10 observations  
## predicted class=1 expected loss=0.4 P(node) =0.006944444  
## class counts: 4 6  
## probabilities: 0.400 0.600

fancyRpartPlot(tree.inj)

Diagram, box and whisker chart

Description automatically generated with medium confidence

# Naive Bayes  
set.seed(20)  
nb.inj <- naiveBayes(IorD ~., data = trainI, laplace = 1, na.action = na.pass)  
summary(nb.cond)

## Length Class Mode   
## apriori 2 table numeric   
## tables 41 -none- list   
## levels 2 -none- character  
## isnumeric 41 -none- logical   
## call 4 -none- call

## Predictions  
# Decision Tree  
tree.injury.pred <- data.frame(predict(tree.inj, testI))  
  
K2 <- c(1:nrow(tree.injury.pred))  
P2 <- c()  
for (x in K2){  
 if(tree.injury.pred[x,1] > 0.5){  
 P2 <- c(P2, 0)}  
 else {  
 P2 <- c(P2, 1)  
 }  
}  
  
tree.injury.pred$pred <- as.factor(P2)  
  
# DT Confusion matrix  
(tree.injury.conf <- confusionMatrix(tree.injury.pred$pred, testI$IorD))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 852 10  
## 1 15 83  
##   
## Accuracy : 0.974   
## 95% CI : (0.9618, 0.9831)  
## No Information Rate : 0.9031   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8547   
##   
## Mcnemar's Test P-Value : 0.4237   
##   
## Sensitivity : 0.9827   
## Specificity : 0.8925   
## Pos Pred Value : 0.9884   
## Neg Pred Value : 0.8469   
## Prevalence : 0.9031   
## Detection Rate : 0.8875   
## Detection Prevalence : 0.8979   
## Balanced Accuracy : 0.9376   
##   
## 'Positive' Class : 0   
##

# Naive Bayes  
nb.injury.pred <- data.frame(predict(nb.inj, testI))  
# NB Confusion Matrix  
(nb.injury.conf <- confusionMatrix(nb.injury.pred$predict.nb.inj..testI., testI$IorD))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 822 8  
## 1 45 85  
##   
## Accuracy : 0.9448   
## 95% CI : (0.9284, 0.9584)  
## No Information Rate : 0.9031   
## P-Value [Acc > NIR] : 1.807e-06   
##   
## Kappa : 0.7321   
##   
## Mcnemar's Test P-Value : 7.615e-07   
##   
## Sensitivity : 0.9481   
## Specificity : 0.9140   
## Pos Pred Value : 0.9904   
## Neg Pred Value : 0.6538   
## Prevalence : 0.9031   
## Detection Rate : 0.8562   
## Detection Prevalence : 0.8646   
## Balanced Accuracy : 0.9310   
##   
## 'Positive' Class : 0   
##