Severity prediction of aviation incidents

Leonardo Tellaroli

15/5/2020

Contents

Abstract

In this document we analyze a dataset containing a list of airplane accident reports and we will try to apply machine learning methods to predict the severity of the consequences of the accidents from the predictors available in a publicly available dataset found on on kaggle .

1 Introduction

1.1 Motivation

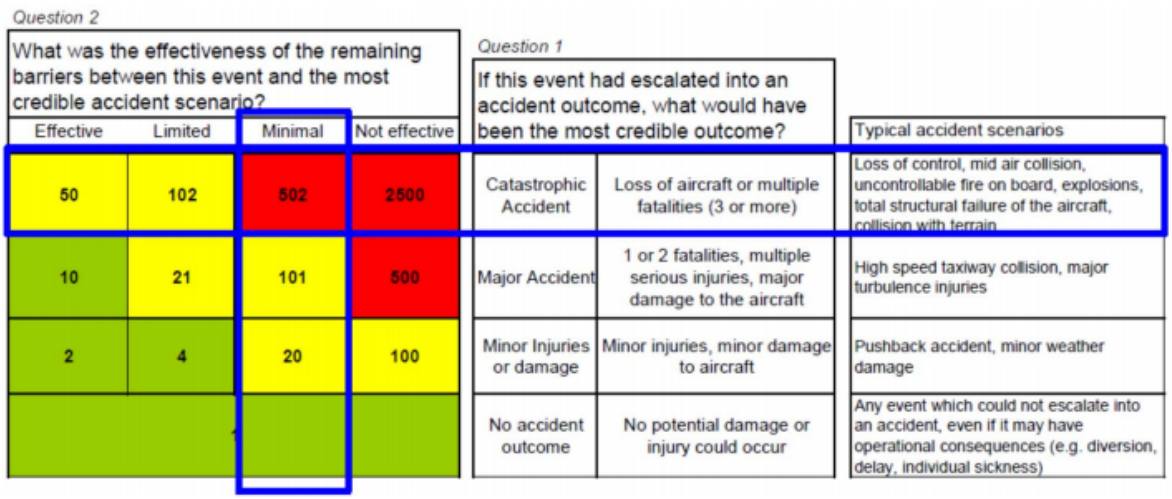
One of the most frequent tasks carried out by Aircraft Operators (Airlines, Aerotaxi, Aerial Work companies) Safety Departments is the classification of the safety reports submitted by crews.
Those safety reports are usually submitted trough an ad hoc web based safety reporting system, which is mandatory in order to have an Air Operator Certificate, and are then classified with standard taxonomies to perform data analysis in order to keep under control the accidents trends.
The submission of those reports is mandatory when the pilots think that the situation they experienced could have developed into an accident.

A crucial step of this process is the assignation of a risk score to each report to assess the damage potential of each near-miss reported had it developed into an accident. This risk score will then be used to prioritize investigations and corrective actions on reports with a higher damage potential.

This risk score is usually calculated with the ARMS-Event Risk Classification methodology, which is thoroughly described here.
In short to use this methodology the Safety Officer tasked with the management of the safety reporting system has to read the report received and find:

- The number and effectiveness of safety barriers available to stop the near-miss development into an accident.
- The potential severity (extent of damage to people and aircraft) of the accident that could have happened.

After the two assessments above the following matrix is used to find the risk score:



Question 2: What was the effectiveness of the remaining barriers between this event and the most credible accident scenario?				Question 1: If this event had escalated into an accident outcome, what would have been the most credible outcome?		Typical accident scenarios
Effective	Limited	Minimal	Not effective			
50	102	502	2500	Catastrophic Accident	Loss of aircraft or multiple fatalities (3 or more)	Loss of control, mid air collision, uncontrollable fire on board, explosions, total structural failure of the aircraft, collision with terrain
10	21	101	500	Major Accident	1 or 2 fatalities, multiple serious injuries, major damage to the aircraft	High speed taxiway collision, major turbulence injuries
2	4	20	100	Minor Injuries or damage	Minor injuries, minor damage to aircraft	Pushback accident, minor weather damage
				No accident outcome	No potential damage or injury could occur	Any event which could not escalate into an accident, even if it may have operational consequences (e.g. diversion, delay, individual sickness)

Figure 1: ERC matrix

The problems of this process are that is quite repetitive, it can take a significant amount of time if a lot of report are received and is highly dependent on the safety officer's personal judgment.
If this process is used by more than one safety investigator in the company a personal bias is clearly noticed and the ERC score assigned will not be consistent sometimes.
To enhance this process we will try to build a machine learning algorithm capable of performing the potential severity assessment.

1.2 Available Data

To develop this algorithm we will use the Airplane Accident Severity dataset available on kaggle.
This dataset contains a list of aviation accidents with their associated Severity. The logic behind the choice of using this dataset is that we will train our model on accidents that actually happened, where the severity of the consequences can be directly observed and is less dependent on an investigator's judgment. This data is also publicly available and ready for machine learning analysis.

2 Data Exploration

2.1 Structure of the dataset

The dataset we chose reports the Severity and 10 predictors related to 1000 airplane accidents plus a unique accident ID to identify each accident.
The accidents are stored as rows with the predictors, Severity and Accident ID as columns of the dataframe. From a preliminary look at the dataframe we see that Severity Levels are stored as a factor with the following 4 levels: Catastrophic, Dangerous, Major, Minor

The available predictors are:

- **Safety_Score**: A measure of how safe the airplane was deemed to be, ranging from 100 to 0
- **Days_Since_Inspection**: days elapsed since the last maintenance inspection, ranging from 23 to 1
- **Total_Safety_Complaints**: number of safety related complaints from the company employees, ranging from 54 to 0
- **Control Metric**: an estimation of how much control the pilot had during the accident, ranging from 100 to 0
- **Turbulence_In_gforces**: turbulence experience during the accident, ranging from 0.883 to 0.134
- **Cabin Temperature**: last recorded cabin temperature, ranging from 97.51 to 74.74
- **Accident_Type_Code**: factor reporting a numeric code related to the type of the accident
- **Max_Elevation**: maximum altitude reached by the flight, ranging from 6.43×10^4 to 831.696
- **Violations**: number of procedures violations during the event, ranging from 5 to 0
- **Adverse_Weather_Metric**: numeric value related to the how challenging the weather was, ranging from 2.365 to 3.164×10^{-4}

We can also see the structure of the dataframe by printing the first 6 rows of the data:

```
## # A tibble: 6 x 12
##   Severity Safety_Score Days_Since_Insp~ Total_Safety_Co~ Control_Metric
##   <ord>           <dbl>            <int>            <int>          <dbl>
## 1 Minor            49.2               14               22           71.3
## 2 Minor            62.5               10               27           72.3
## 3 Dangero~         63.1               13               16           66.4
## 4 Major            48.1               11                9           74.7
## 5 Dangero~         26.5               13               25           47.9
## 6 Minor            43.3               15                0           73.3
## # ... with 7 more variables: Turbulence_In_gforces <dbl>,
## #   Cabin_Temperature <dbl>, Accident_Type_Code <fct>, Max_Elevation <dbl>,
## #   Violations <int>, Adverse_Weather_Metric <dbl>, Accident_ID <int>
```

2.2 Data Wrangling

The data is provided as a Comma Separated Value(.csv) file so we first have to import in with the *read.csv* function:

```
raw_data <- read.csv(dl, header = TRUE, stringsAsFactors = FALSE)
```

Then we have to convert the Severity and Accident_Type_ID into factors because they are categorical variables.
We store the Severity value that we want to predict as an ordered factor and relabel it using *Catastrophic*, *Dangerous*, *Major* and *Minor* because those names are more frequently used in the aviation industry.

```
# convert severity and accident type as factors
accidents <- raw_data %>% mutate(Severity = factor(Severity),
    Accident_Type_Code = factor(Accident_Type_Code))
# get the severity levels and converts sverity into an
# ordered factor
severity_levels <- levels(accidents$Severity)
accidents <- accidents %>% mutate(Severity = ordered(as.character(Severity),
    levels = c(severity_levels[1], severity_levels[3], severity_levels[4],
        severity_levels[2]), labels = c("Catastrophic", "Dangerous",
        "Major", "Minor")))
```

We also check the number of NA values in the data and we see that NAs are not present.

```
print(c("Numbers of NAs in the data equal to", sum(is.na(accidents))))
```

```
## [1] "Numbers of NAs in the data equal to" "0"
```

We now have to split the data into training ad test set. Because we don't know if the process we are considering can be successfully carried out automatically, we chose to use a larger test set than usual, including 20%of the data in it and leaving 80% of the data for the training set.
We use the *createDataPartition* function from the **Caret** package to split the data:

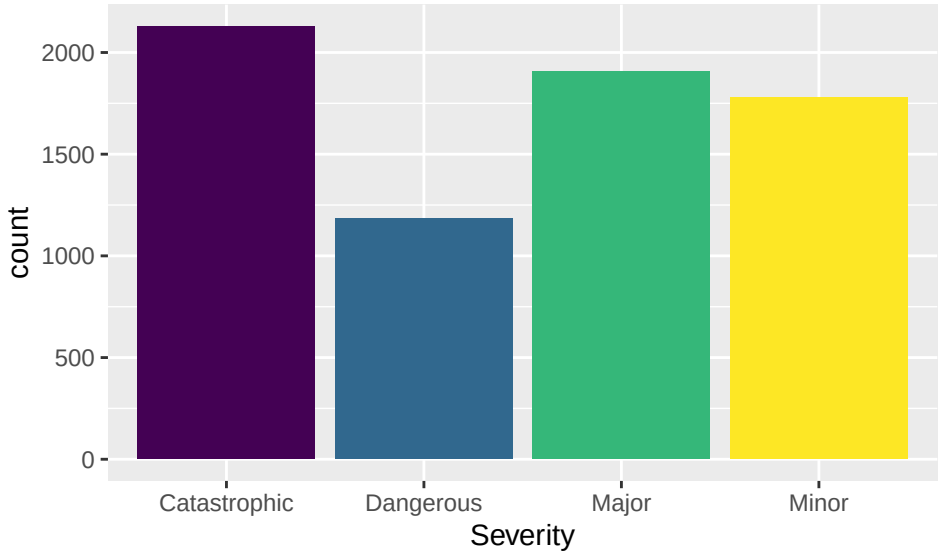
```
index <- createDataPartition(y = accidents$Safety_Score, times = 1,
    p = 0.3, list = FALSE)
test_set <- accidents[index, ]
train_set <- accidents[-index, ]
```

2.3 Data visualization

Now we start to study our data to find characteristics that will be useful when developing the prediction algorithms. This part of the analysis will be performed using data coming only from the training set.

We first assess the proportions of severity Outcomes in our data and plot the number of reports related to each severity category:

```
## Catastrophic    Dangerous        Major        Minor
##        0.304        0.169        0.272        0.254
```

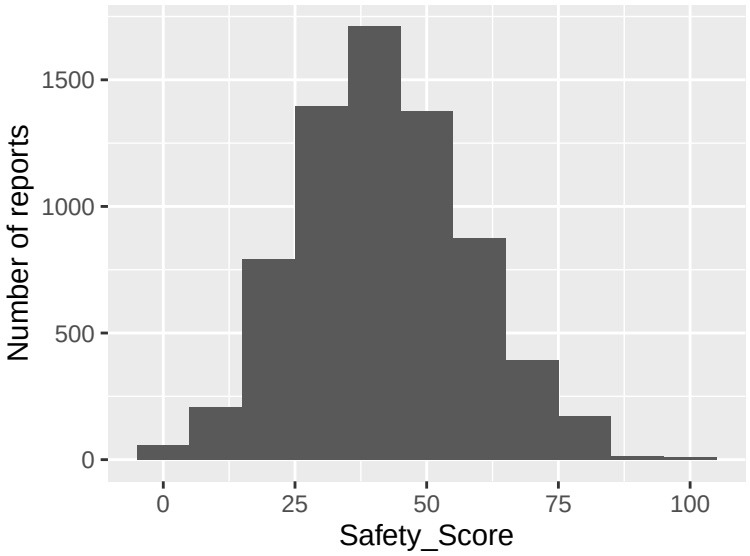


We see that the *Dangerous* class has a lower number of related accidents, but the other classes are quite balanced.

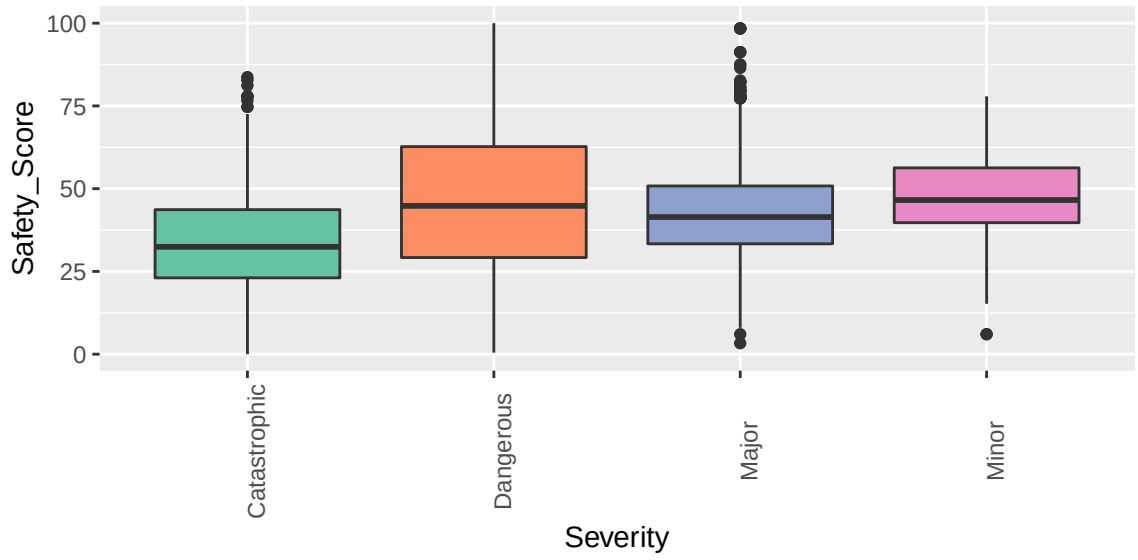
We continue the analysis visualizing the effect that the various predictors have on the Severity.

Safety Score

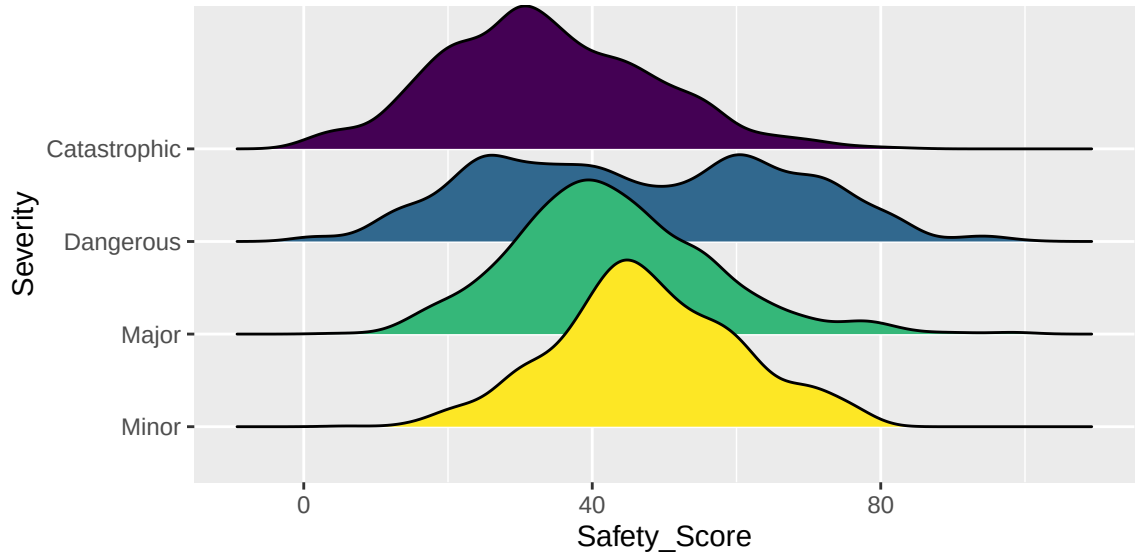
We first plot the distribution of Safety Scores across our training data:



Then we visualize boxplots of the safety score grouped by Severity of occurrences and the density plots of Safety Scores grouped by severity:

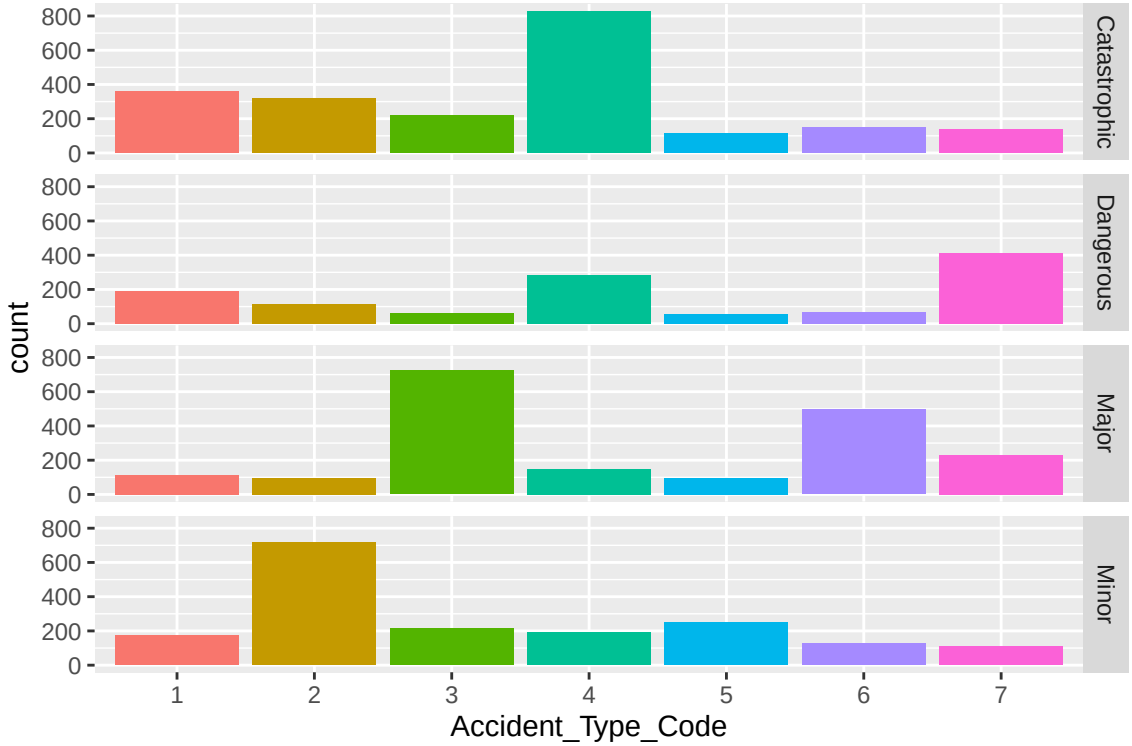



```
## Picking joint bandwidth of 3.08
```

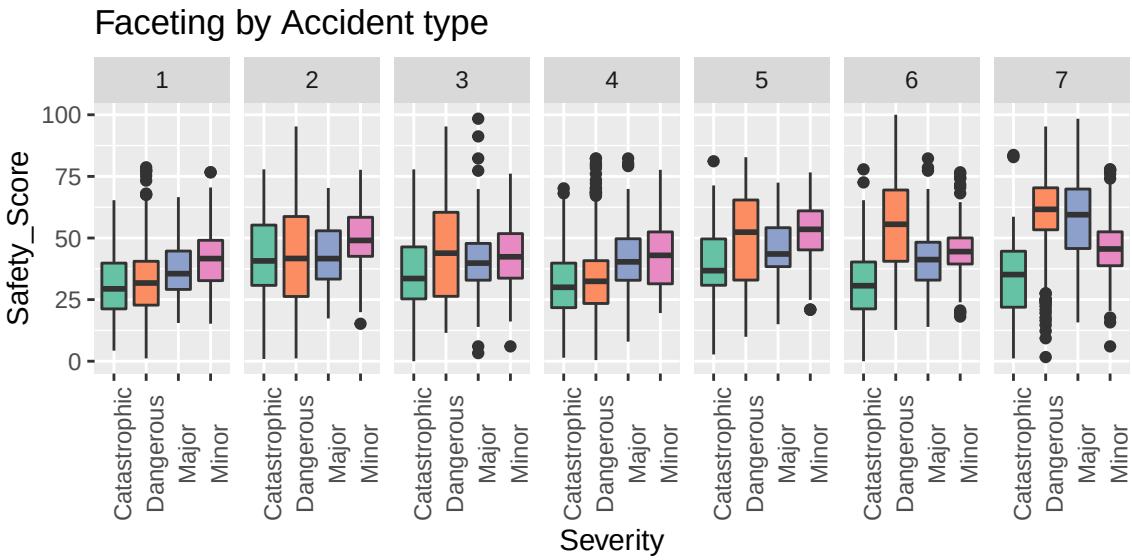


We see that the mean safety score is quite related with the severity but the distributions are too widespread to really predict severity from Safety Score only.
The shapes of the safety scores does not show a significant severity to severity variability except for the *Dangerous* class which is differenlty distributed.

We then visualize the number of reports related to each accident type faceting by severity:



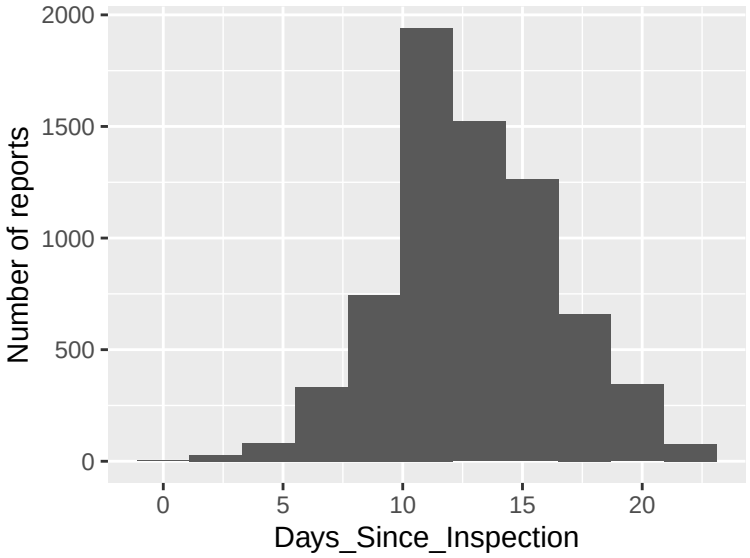
We see that for some Severity outcomes certain accident types are much more frequent than others.
We now repeat the previous boxplot of Safety Score vs Severity faceting by accident type:



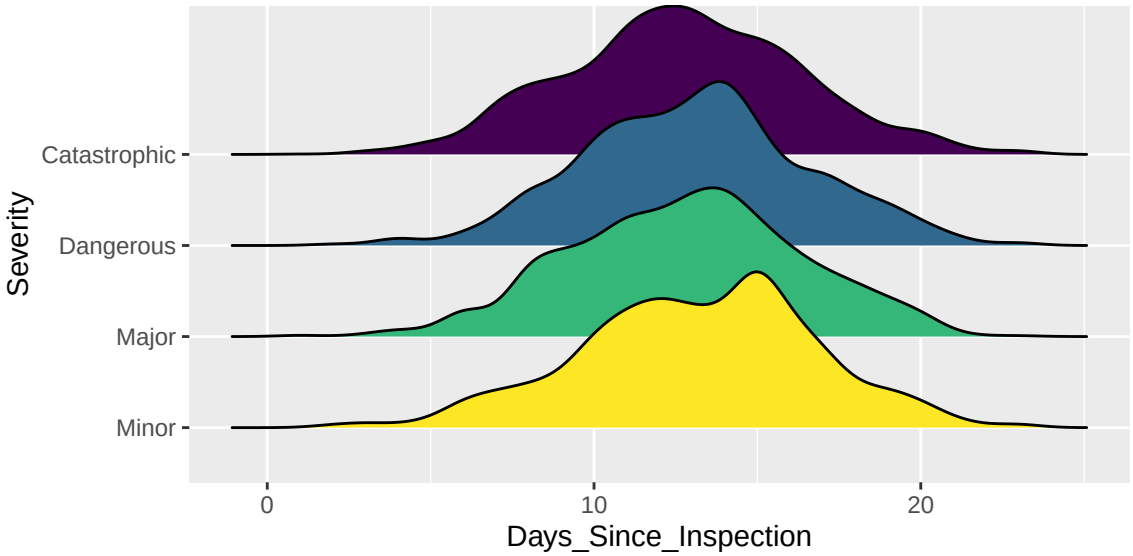
With this grouping by Accident type we see that difference in Safety Score means related to the Severity of the outcome are more significant.
A successful algorithm will thus need to be able to classify the data by accident type.

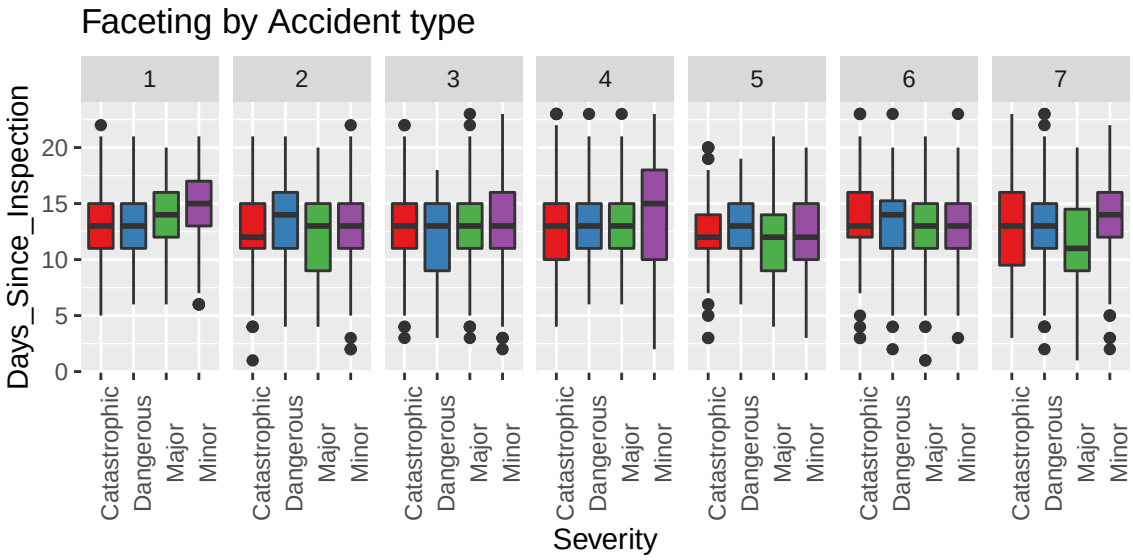
Days since inspection

We repay the same process for the days since inspection variable:




```
## Picking joint bandwidth of 0.693
```

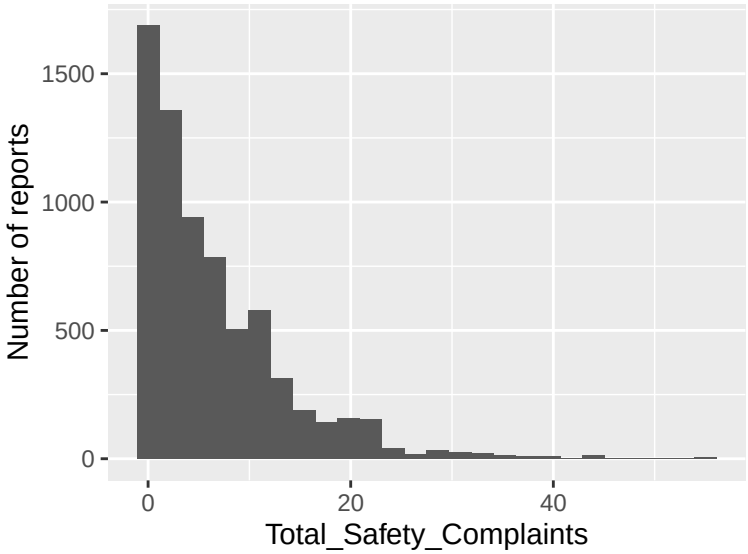




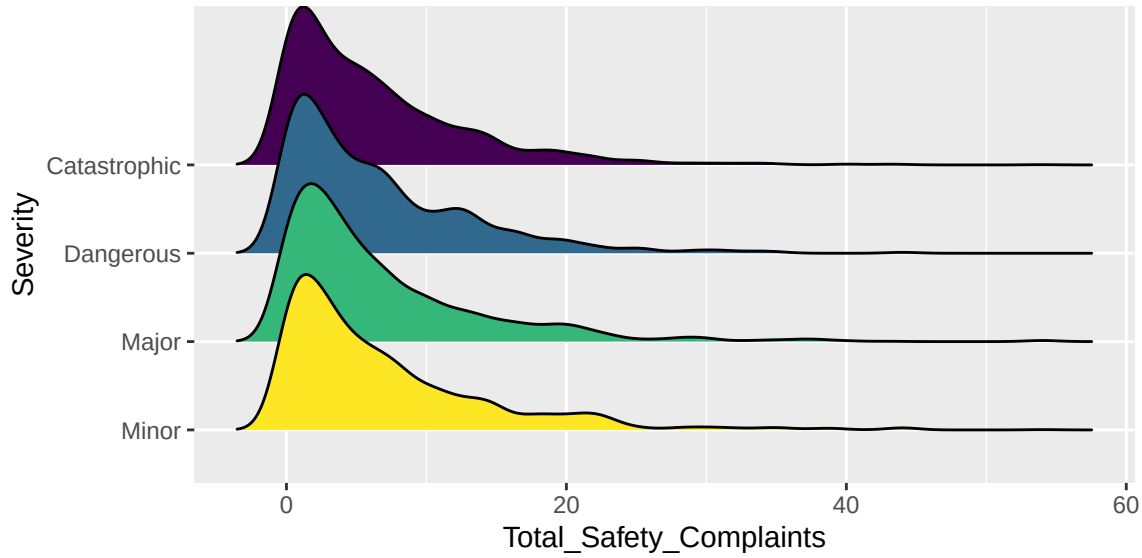
We see that this variable does not show much variability across various severity levels, even when faceted by accident type.

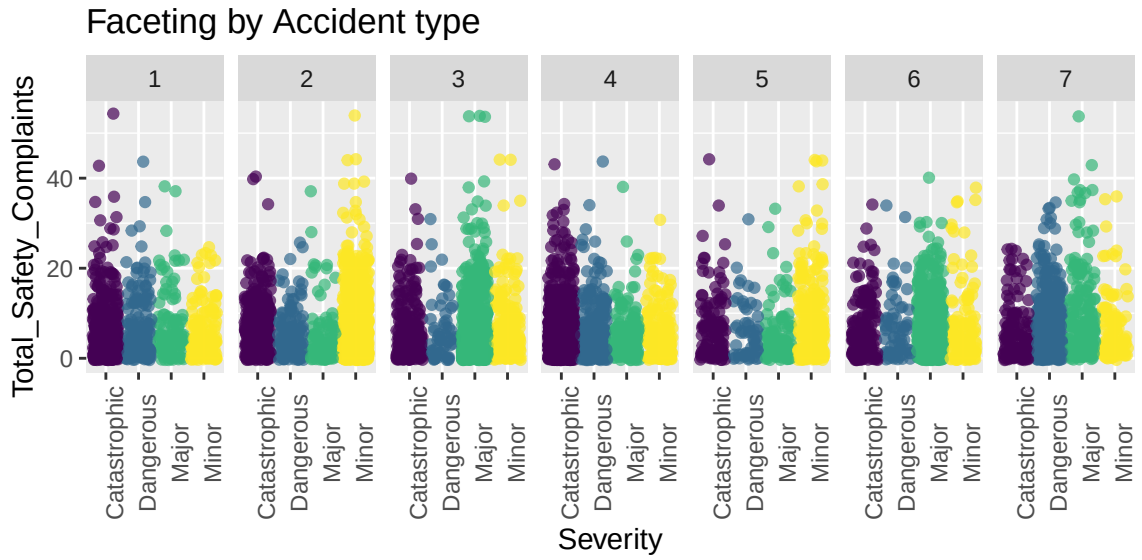
Safety Complaints

The previous plots will be now created using Safety Complaints, because the distribution of this variable is skewed to the left we chose to use scatterplots instead of boxplots:




```
## Picking joint bandwidth of 1.18
```

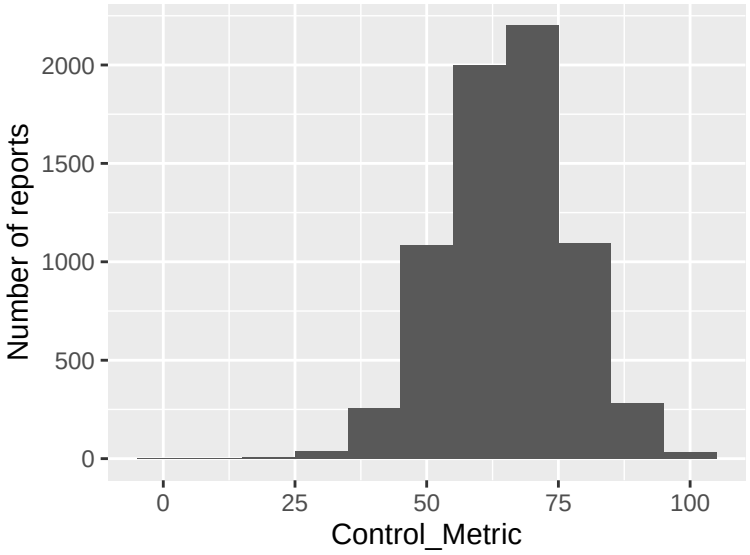




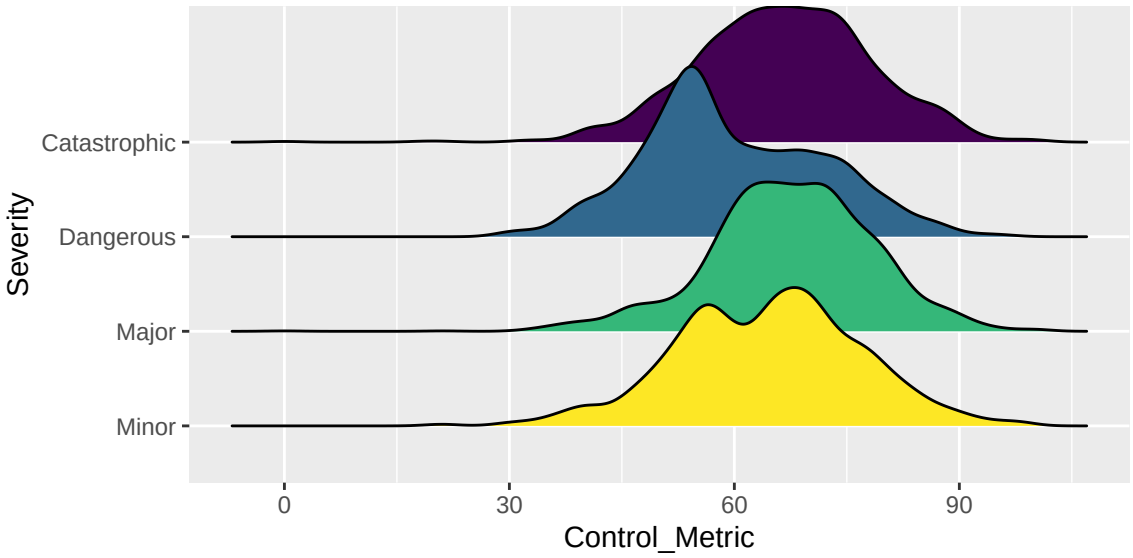
Also for this variable we see that there are no really significant differences across various severity levels.

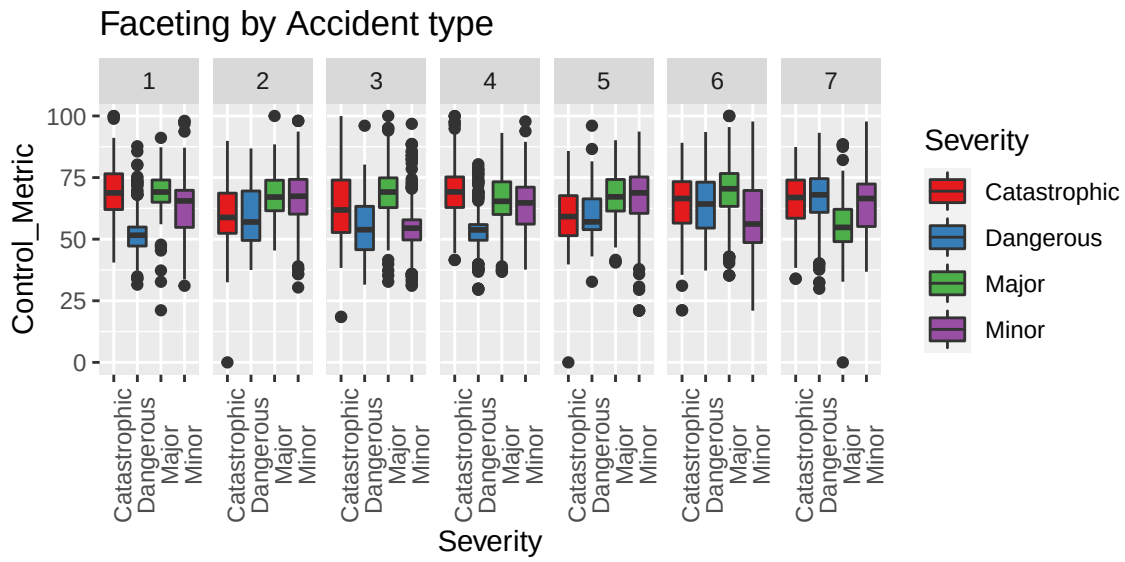
Control Metric

The analysis continues considering the control metric variable:




```
## Picking joint bandwidth of 2.33
```

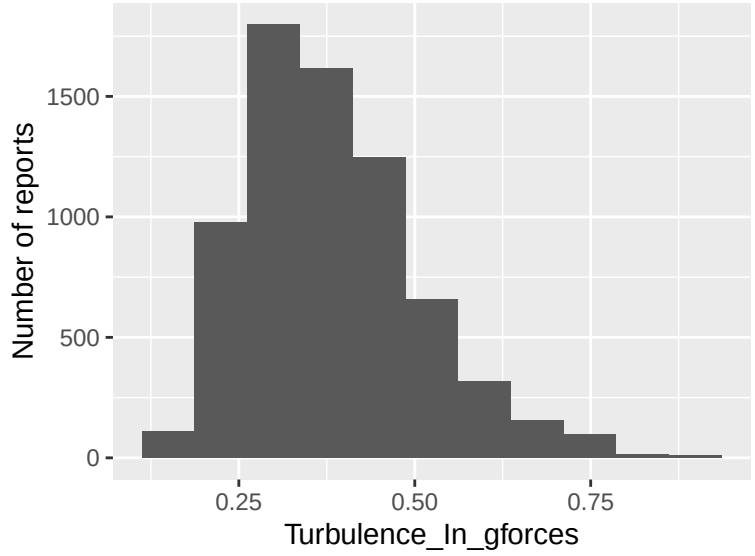




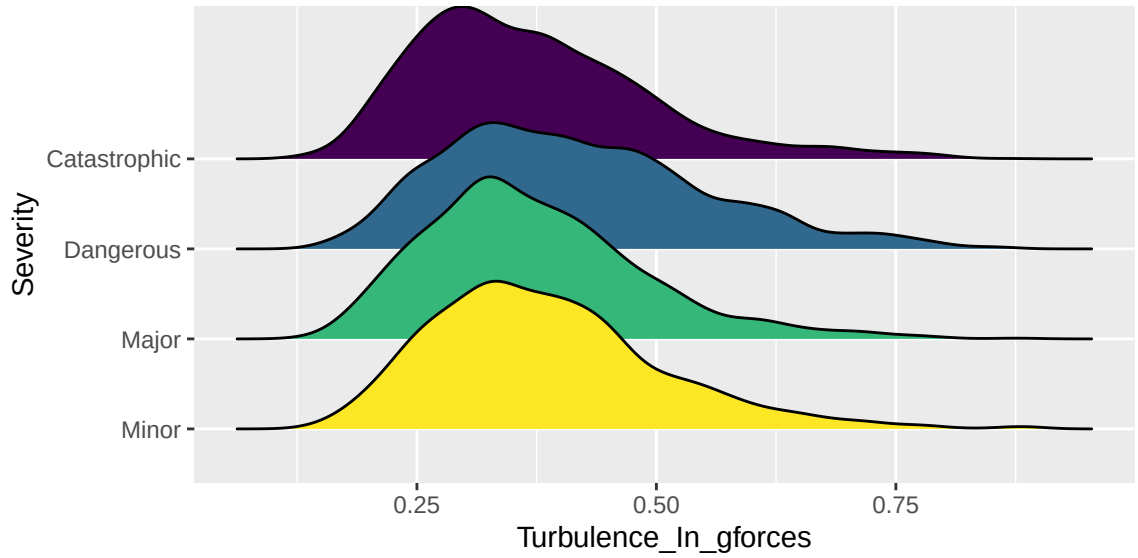
We see that this variable shows more significant differences useful to predict severity but the boxplots continue to have significant intersections, so we will not be able to build our predictions completely on this variable.

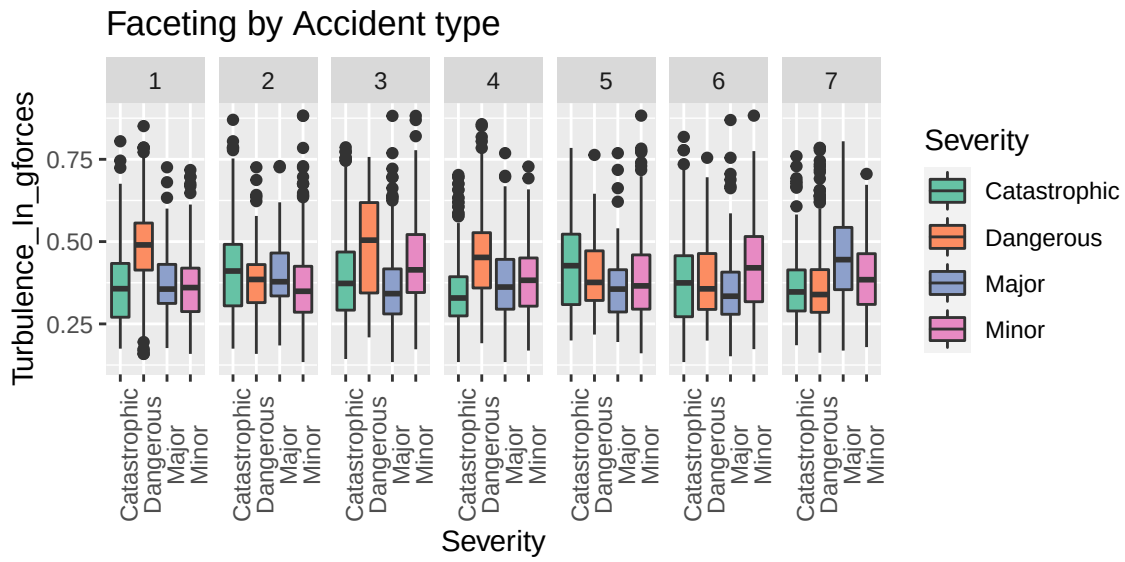
Turbulence

We go on investigating the distribution of turbulence:




```
## Picking joint bandwidth of 0.0239
```

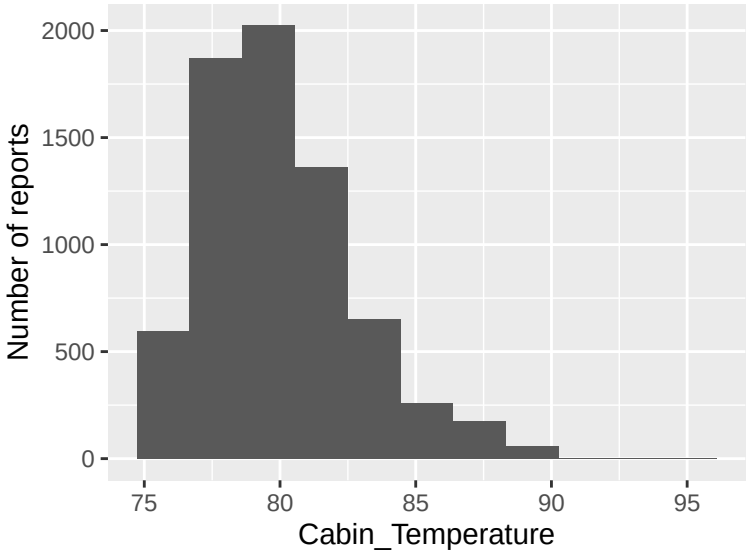




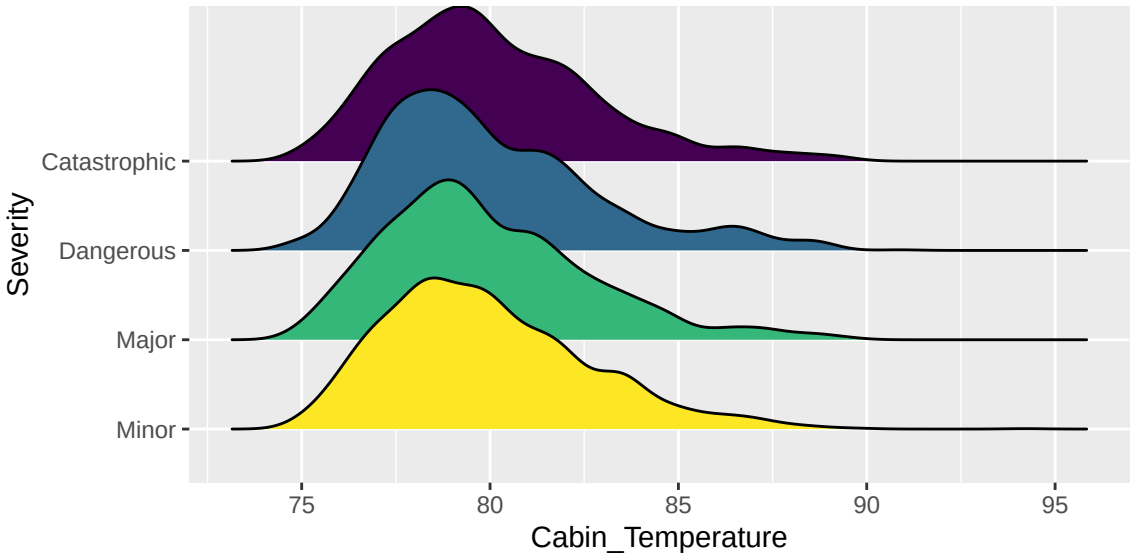
The situation of this variable is similar to the previous one, there are differences but they are not significant enough to be used as the only base for our predictions.

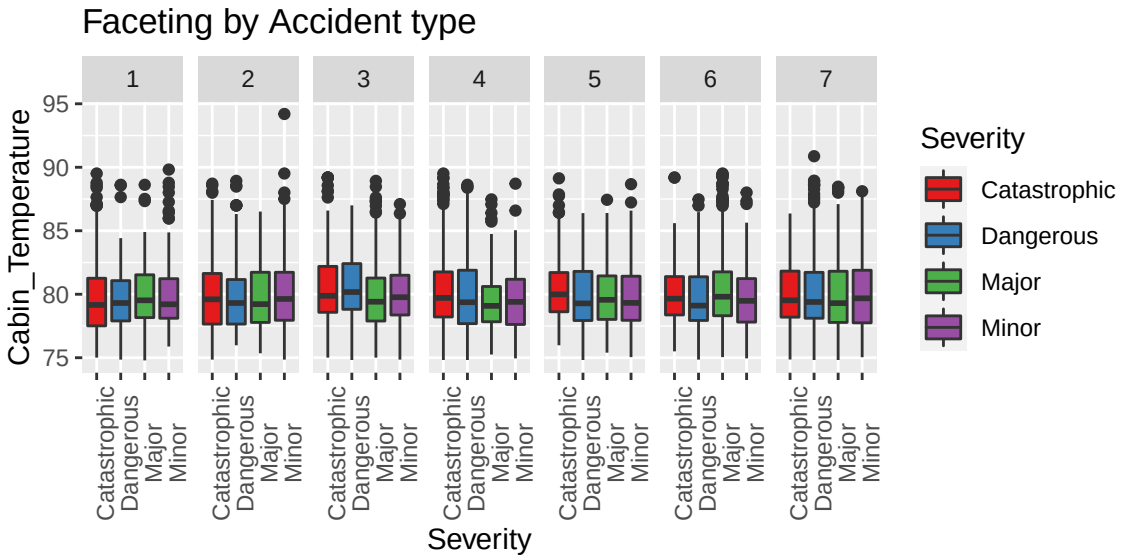
Cabin Temperature

We now repeat the process with Cabin Temperature:



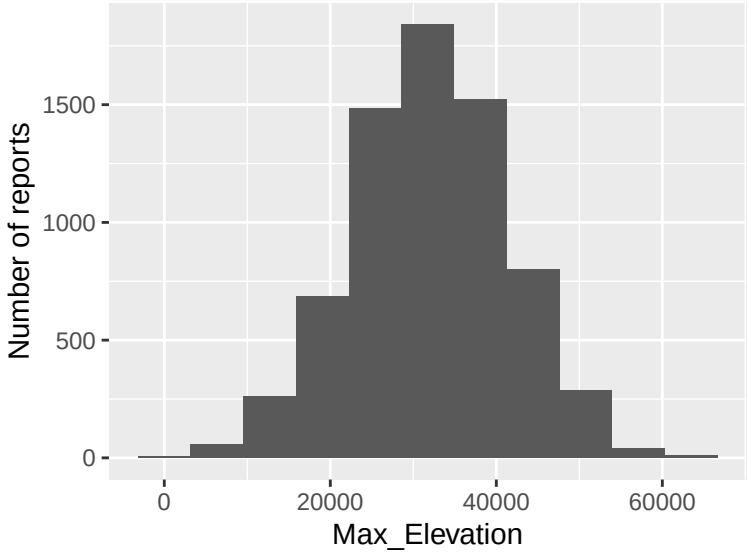

```
## Picking joint bandwidth of 0.546
```



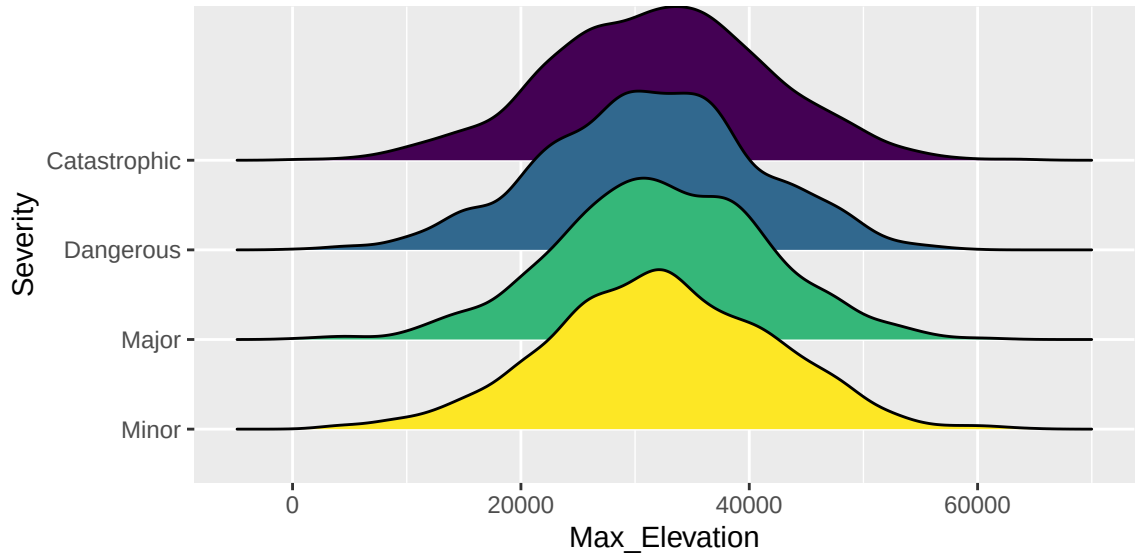


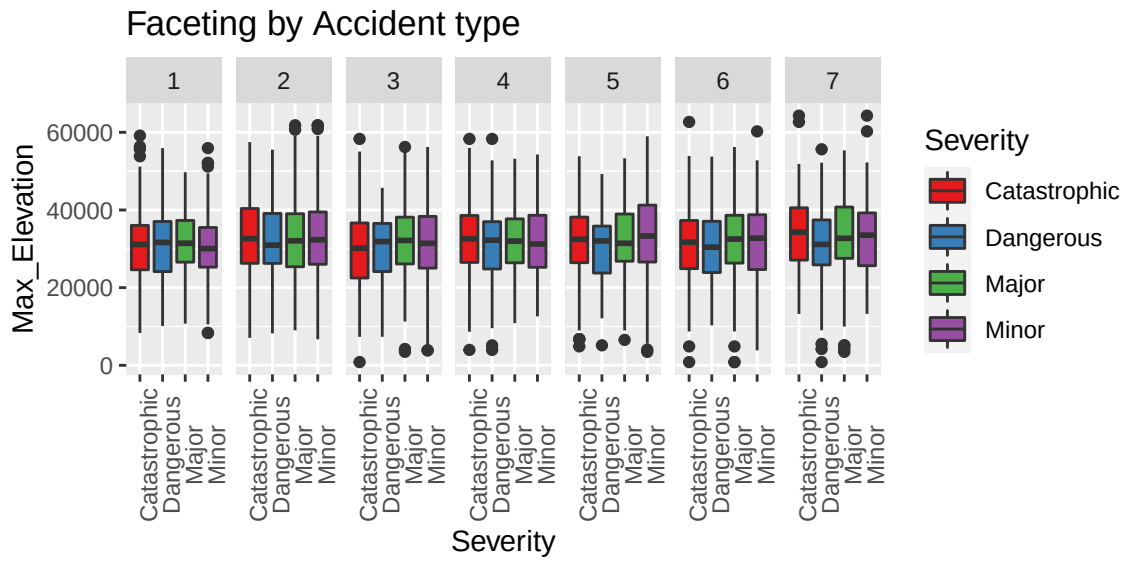
We see that for this variable there is almost no variability related to accident severity, we will thus avoid using it in our prediction models.

Max Altitude Distribution




```
## Picking joint bandwidth of 1900
```

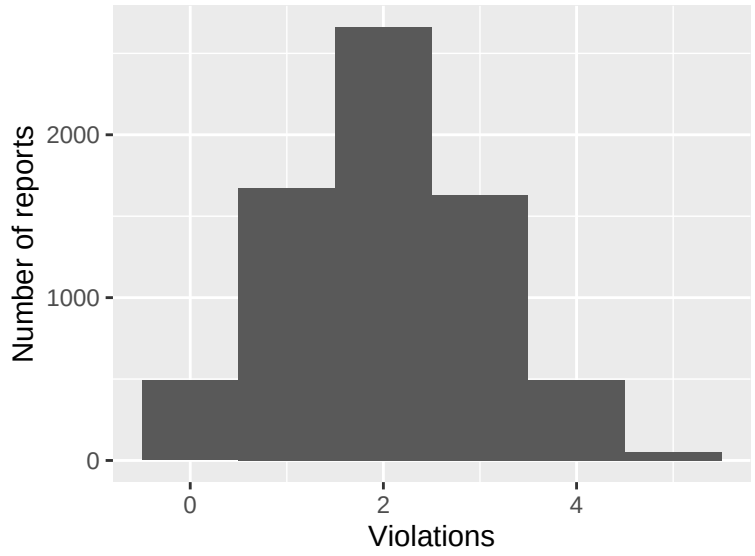




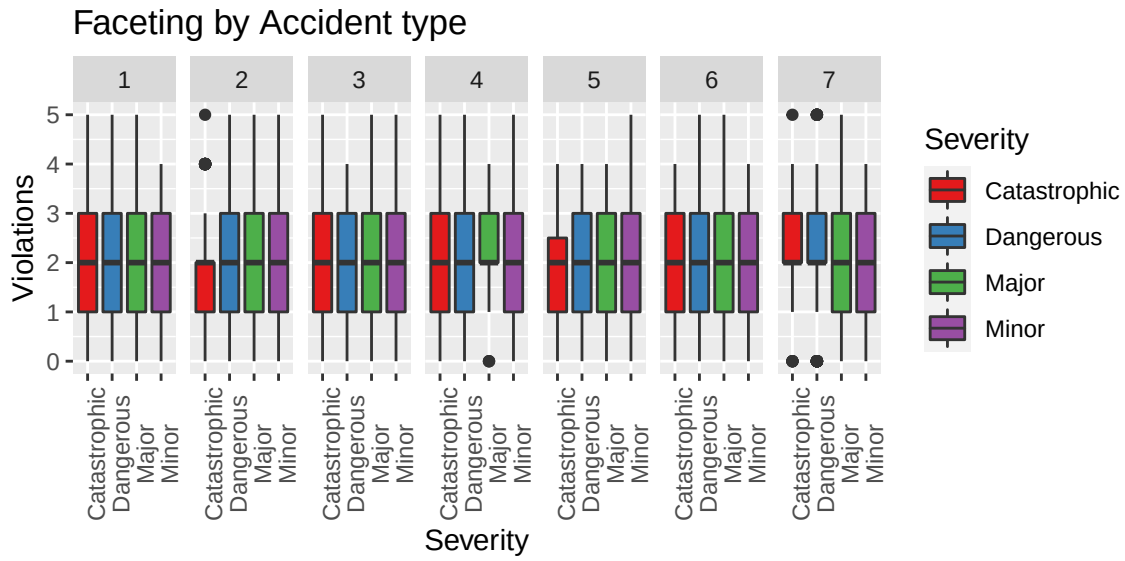
Even for maximum altitude distribution we see that there is not enough severity to severity variability.

Number of Violations

Here we see the distribution of the number of violations:



And the related boxlots grouped by Severity and faceted by accident type



Even the number of violations does not give us enough information for basing our predictions on this variable.

Variable vs Variable plots

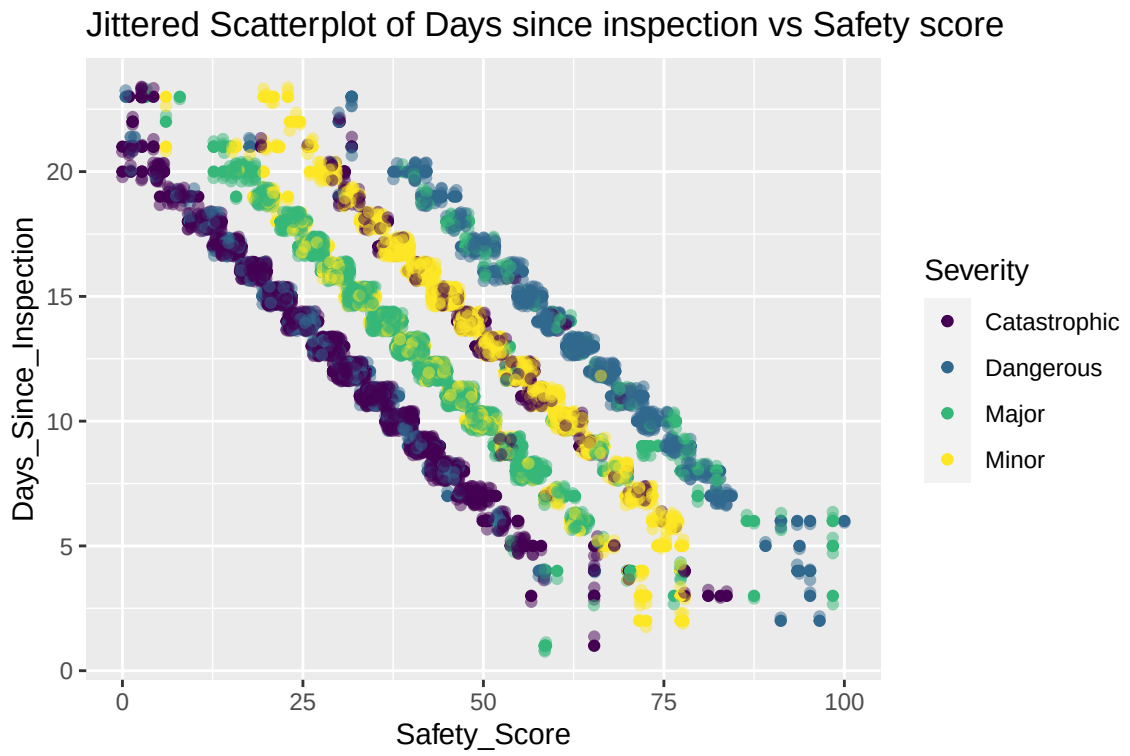
We saw that no variable has enough severity to severity variability to build a useful prediction model, we will thus continue our analysis searching for Severity clusters in variable vs variable plots to find two dimensional relationships between variables and severity.

To plot the high number of plots needed to explore the correlations between each pair of variables we decided to use a for cycle to automatize the plot creations, storing all the ggplot objects created in a list:

```
# initialize list to store following plots
plots_list <- vector(mode = "list")
```

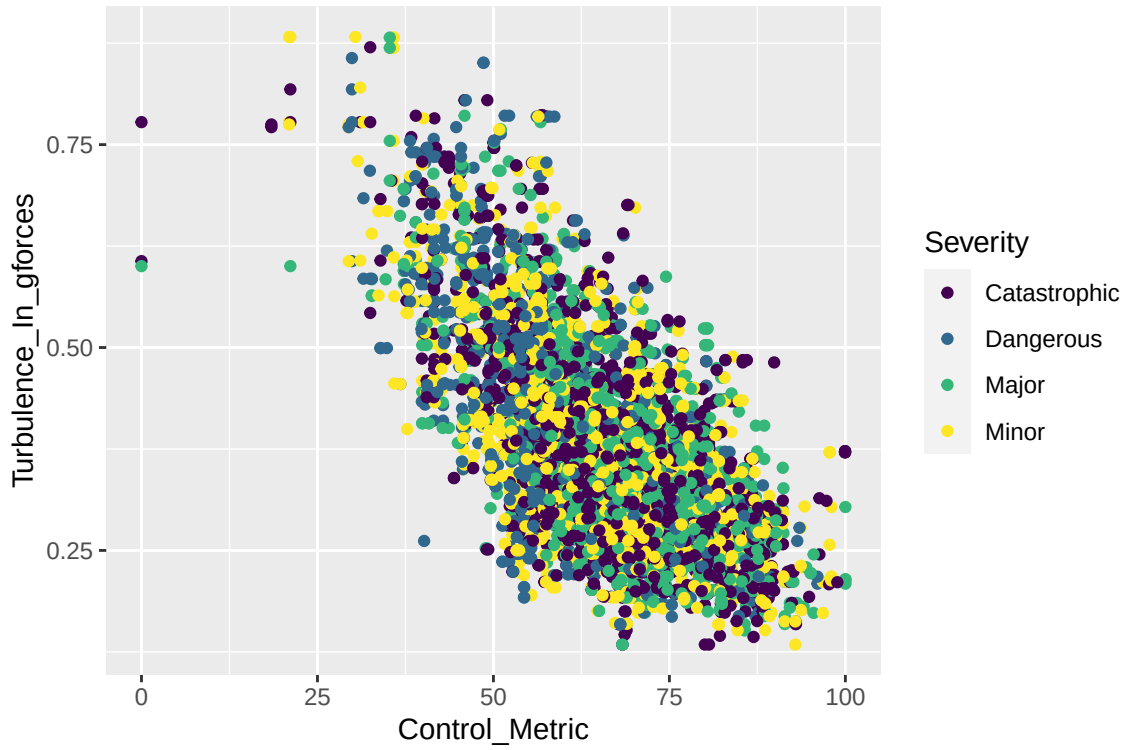
```
# for cycle to create scatterplots for each variable to
# variable combination, setting point's color based on the
# Severity, to search for clusters of data
for (i in seq(1:length(colnames(train_set)))) {
    plots_list2 <- vector(mode = "list")
    for (j in seq(1:length(colnames(train_set)))) {
        xvalue = train_set[, i]
        yvalue = train_set[, j]
        plotdata <- data.frame(xvalue, yvalue, Severity = train_set$Severity,
            stringsAsFactors = FALSE)
        plots_list2[[j]] <- plotdata %>% ggplot(aes(x = xvalue,
            y = yvalue, color = Severity)) + geom_point() + xlab(colnames(test_set)[i]) +
            ylab(colnames(test_set)[j])
    }
    names(plots_list2) <- colnames(train_set)
    plots_list[[i]] <- plots_list2
}
names(plots_list) <- colnames(train_set)
```

Looking the various plots we see interesting features in the safety score vs days since inspection plot that we report here after adding jitter to better visualize the clusters:

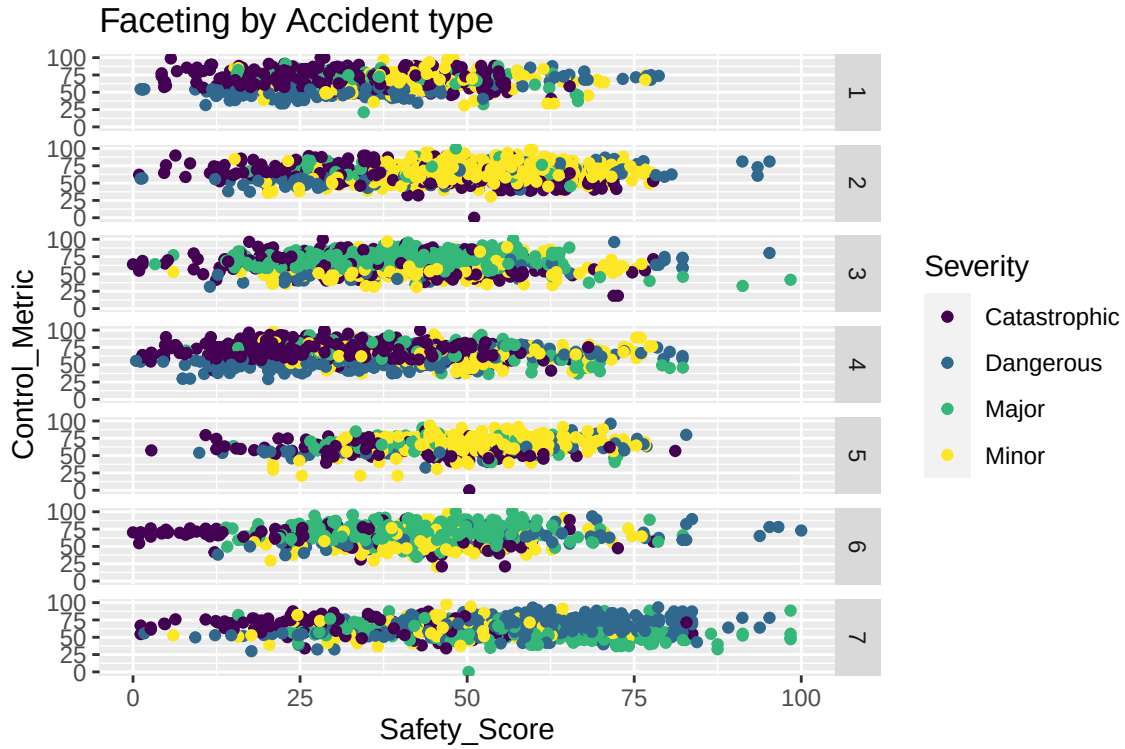


In this plot we see clear clusters of data with the same severity outcome at certain Safety_score - Days_Since_Inspection combinations.

In another plot, the control metric vs turbulence plot, we don't see useful clusters but we visualize a clear negative correlation between those two variables.
This negative correlations is expected since turbulence makes the control of an aircraft more challenging.

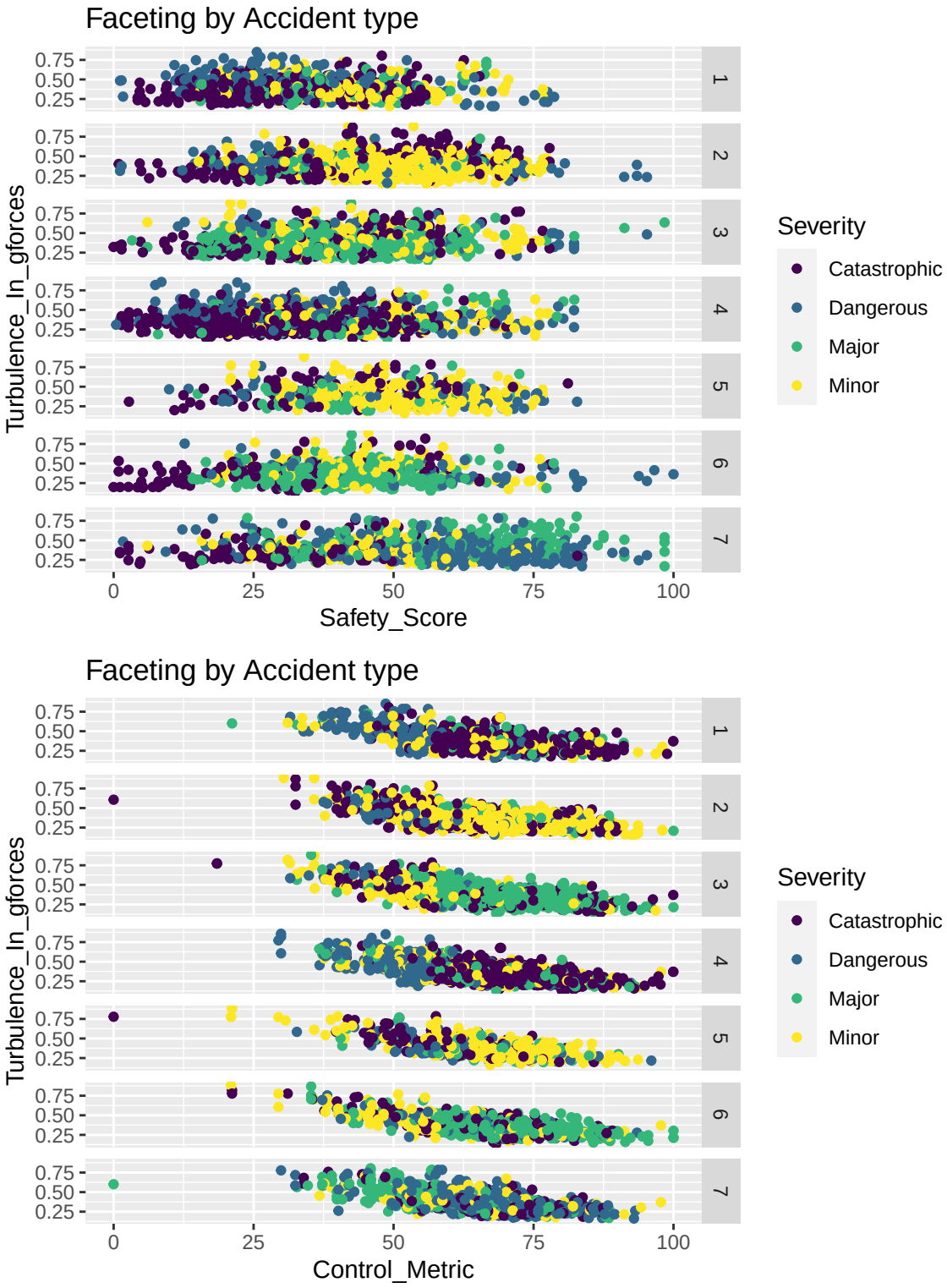


We repeat the previous process introducing faceting by accident type, to search for other interesting clusters. In the Control Metric vs Safety Score plot we see some different clusters across the various accident types:



A similar situation of different clusters in different accident types groups can be seen also in the Turbulence

vs Safety score plot and the Turbulence vs Control Metric plot:

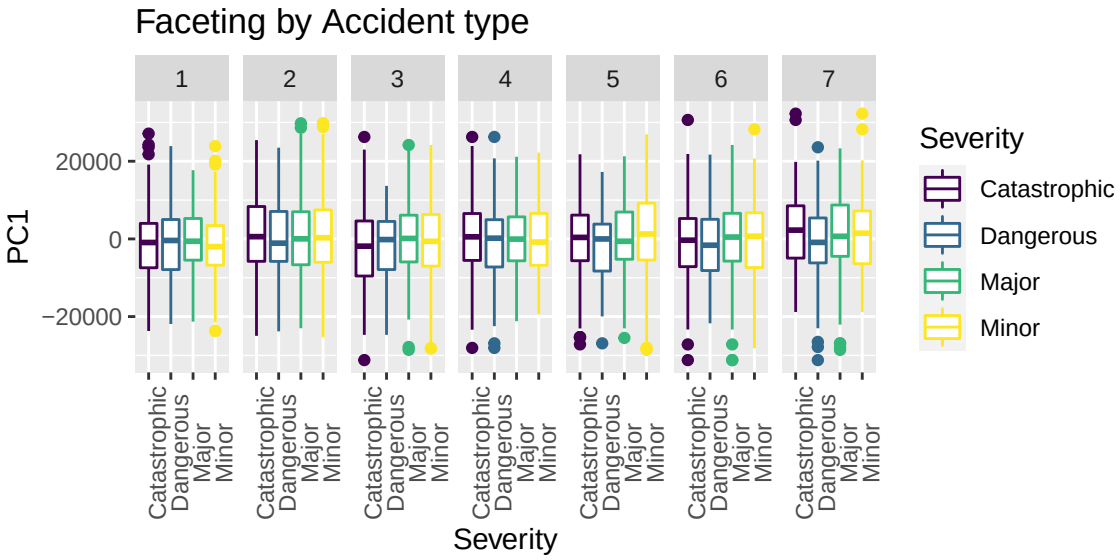


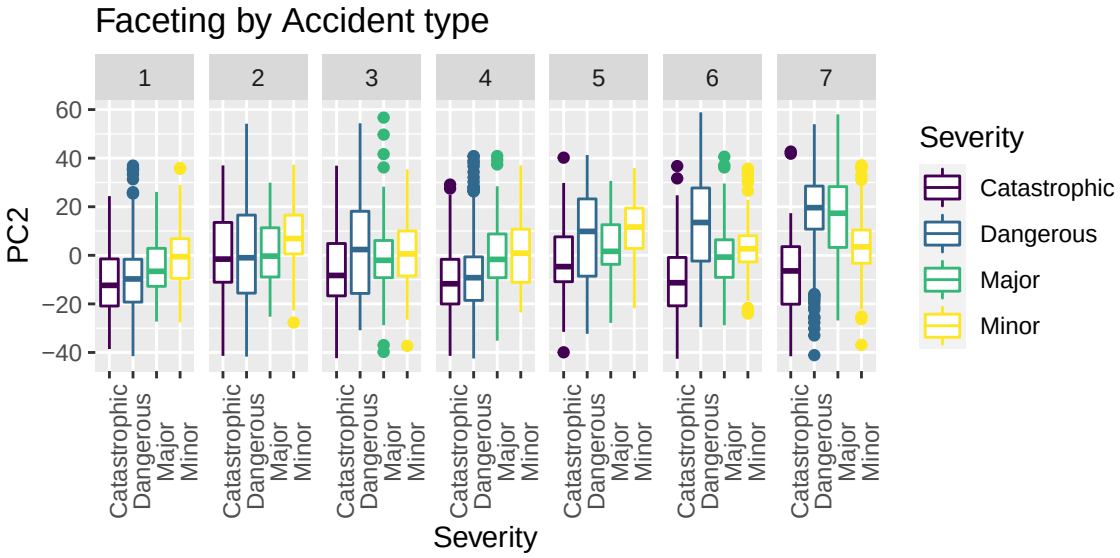
Visualizing principal components

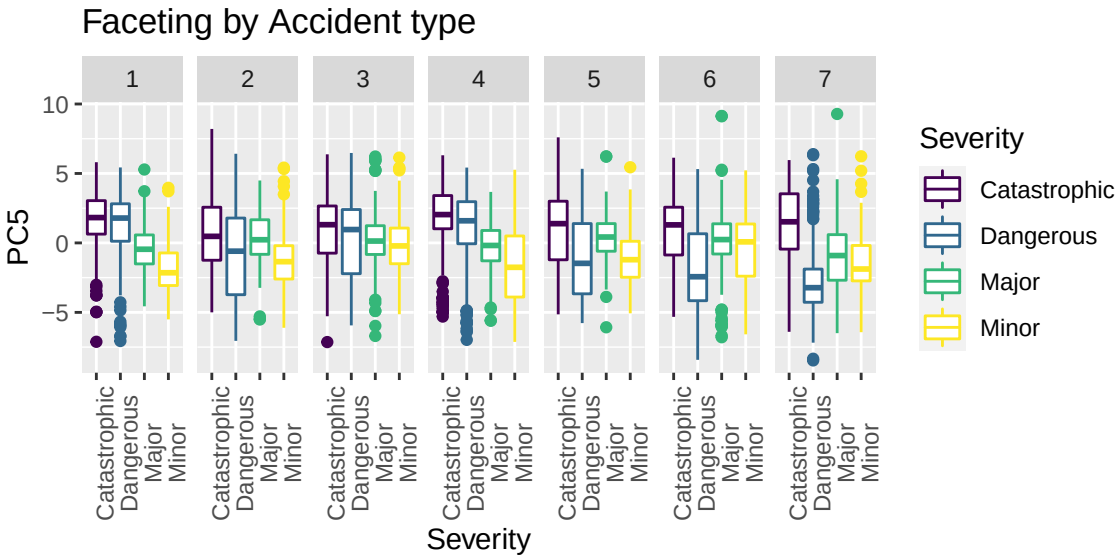
We continue our data exploration by converting the train_set to a matrix, excluding factor variables and then applyingPrincipal Component Analysis on this matrix:

```
# creating a matrix containing only numeric predictor as
# columms, not including Severity, accidentID, Accident type
# and cabin temperature (we have seen that cabin temperature
# boxplots are almost equal for differents levels of
# severity)
predictors_matrix <- as.matrix(train_set %>% select(-Severity,
    -Accident_ID, -Accident_Type_Code, -Cabin_Temperature))
# computing pca
pca <- prcomp(predictors_matrix)
```

We then plot boxplots of principal components vs severity, faceting by accident type.
Here we report some of the plots:







We see that the first principal component, even if it accounts for most of the data variability, does not show variability across severity and accident type values.
Other principal components like PC2 and PC5 show more useful variability for our goal, but it's still not enough to build a prediction algorithm over those principal components.

3 Model development

3.1 Random sampling

To have a benchmark to evaluate performance of the next models we start by computing the accuracy of a prediction created by randomly sampling a Severity outcome for each row of the test set:

```
random_severity <- ordered(sample(severity_levels, nrow(test_set),
    replace = TRUE))  #samples severity predictions randomly
# we then compute the confusion matrix of the random
# prediction
confMatrix_random <- confusionMatrix(random_severity, reference = test_set$Severity)
```

The resulting accuracy is equal to 0.254 . As expected this accuracy value is quite poor.

3.2 Random sampling with probabilities from the training set

To have another performance benchmark we enhance the previous random sampler by setting the probability of sampling a certain severity value from the proportions of that severity value in the training set:

```
# Then we try a model that randomly selects severity levels
# using the proportions of severity level in the training set
# as sampling probabilities
proportionsOnly_severity <- ordered(sample(severity_levels, nrow(test_set),
    replace = TRUE, prob = train_set_proportions))
# And we compute the confusion matrix of this model
confMatrix_proportionsOnly <- confusionMatrix(proportionsOnly_severity,
    reference = test_set$Severity)
```

The resulting accuracy is equal to 0.265 . As expected the accuracy value slightly improved but is still poor.

3.3 Predictions using only Accident Type

We now find the most frequent severity value for each one of the 7 accidents types. We do this by grouping the training_set by accident type and then computing the mode of the Severity column for each accident type group.
We get the following result:

```
AccTypeSeverities <- train_set %>% group_by(Accident_Type_Code) %>%
    dplyr::summarize(PredSeverity = as.character(factor_mode(Severity)))

AccTypeSeverities
```

```
## # A tibble: 7 x 2
##   Accident_Type_Code PredSeverity
##   <fct>              <chr>
## 1 1                  Catastrophic
## 2 2                  Minor
## 3 3                  Major
## 4 4                  Catastrophic
## 5 5                  Minor
## 6 6                  Major
## 7 7                  Dangerous
```

```
# factor mode is a custom function to find the mode of a
# factor variable
```

Using the tibble above to associate a predicted severity to each row in the test set we find that the accuracy of this method is equal to 0.543
The accuracy improved substantially compared with random sampling but it is still not acceptable.

3.4 Further data preparation

In preparation for the further implementation of more computationally heavy model we need to do two further data processing steps:

First we remove from the training set the Accident_ID variable, as this is not a predictor but only a unique id associated to each row, then we remove the Cabin Temperature column as we saw in data exploration that this is the least useful predictor having almost no severity to severity variability.

We also notice that in our predictors we have a factor variable, the accident type ID. Machine learning algorithm will treat this variable as numeric but we see that it does not make much sense because computations carried out on this predictor will consider our 7Th accident type as a 7 and our 1St accident type as a 1 when in reality they are only different types, they do not represent the intensity of something.
To avoid error in the computations of the model we need to use **one-hot econding**. This encoding method spreads our accident type column across seven columns (because our factor has 7 levels) each assuming value 1 when the row is related to its corresponding factor level and value 0 when it is not.

We can see how our dataset is transformed below:

```
## # A tibble: 6 x 7
##   Accident_Type_C~ Accident_Type_C~ Accident_Type_C~ Accident_Type_C~
##              <dbl>            <dbl>            <dbl>            <dbl>
## 1                0                1                0                0
## 2                0                1                0                0
## 3                0                0                1                0
## 4                0                0                0                1
## 5                0                0                0                1
## 6                0                1                0                0
```

```
## # ... with 3 more variables: Accident_Type_Code.5 <dbl>,
## #   Accident_Type_Code.6 <dbl>, Accident_Type_Code.7 <dbl>
```

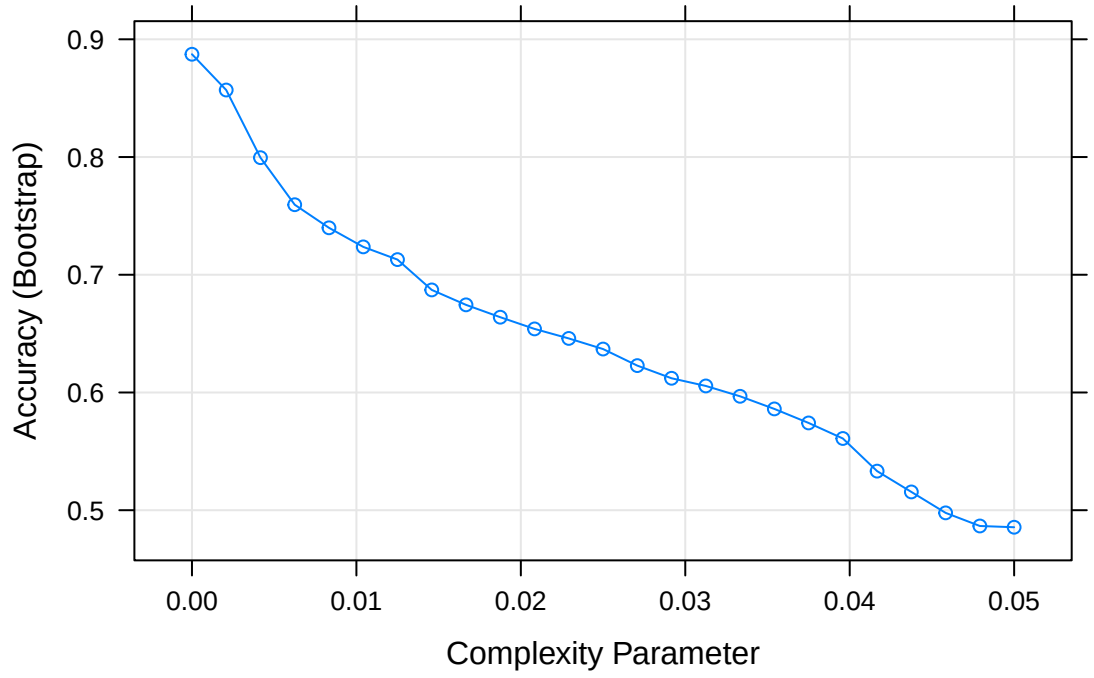
To perform this data processing task we can use the *dummyVars* function from **Caret** and then bind the results of this function with our original dataframe, removing the Accident_Type_Code column:

```
# we split the data contained in accident type code in 7
# columns with only 1 or 0 values to prepare the data for the
# following predictions alghoritms
dummy_test <- data.frame(predict(dummyVars("~ Accident_Type_Code",
    data = test_set), newdata = test_set))
# and we bind the newly created column to our test set,
# deleting the Accidet_Tye_Code column
test_set_dummy <- cbind(test_set %>% select(-Accident_Type_Code),
    dummy_test)
# We repeat the same process for the training set
dummy_train <- data.frame(predict(dummyVars("~ Accident_Type_Code",
    data = train_set), newdata = train_set))
train_set_dummy <- cbind(train_set %>% select(-Accident_Type_Code),
    dummy_train)
```

3.5 Classification Tree

The decision process that we have to automate is a classification problem, involves a factor variable and it is a modelling of a classification process usually carried out by a human operator.
We thus choose to start to take the road of Classification Trees. We will first implement a Classification Three thanks to the **rpart** and **caret** packages, tuning the algorithm over complexity parameter values ranging from 0 to 0.05.

We get the following accuracy vs tuning parameter plot:



We see that the best accuracy is obtained by setting the complexity parameter to 0.
We know that extremely low values of the complexity parameter can lead to overtraining, we thus check the accuracy of this model on the test set.

The accuracy on the bootstrapped training set is equal to:0.887

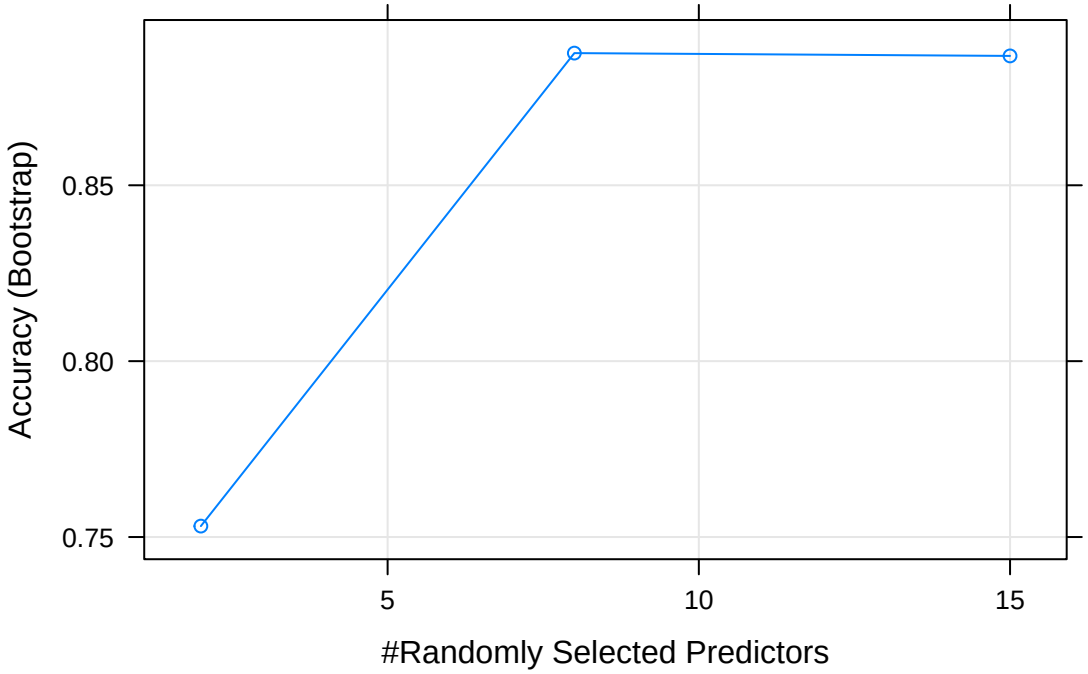
The accuracy computed on the test set is: 0.905

We see that those value are very similar so overtraining has not been a problem in this case.

The computational time required on a 2.4Ghz processor is: 30 seconds

3.6 Random Forest

We now try to implement a Random Forest model trough the **Rborist** package.
We see that the accuracy of this model converges at a relatively low number of random trees used:



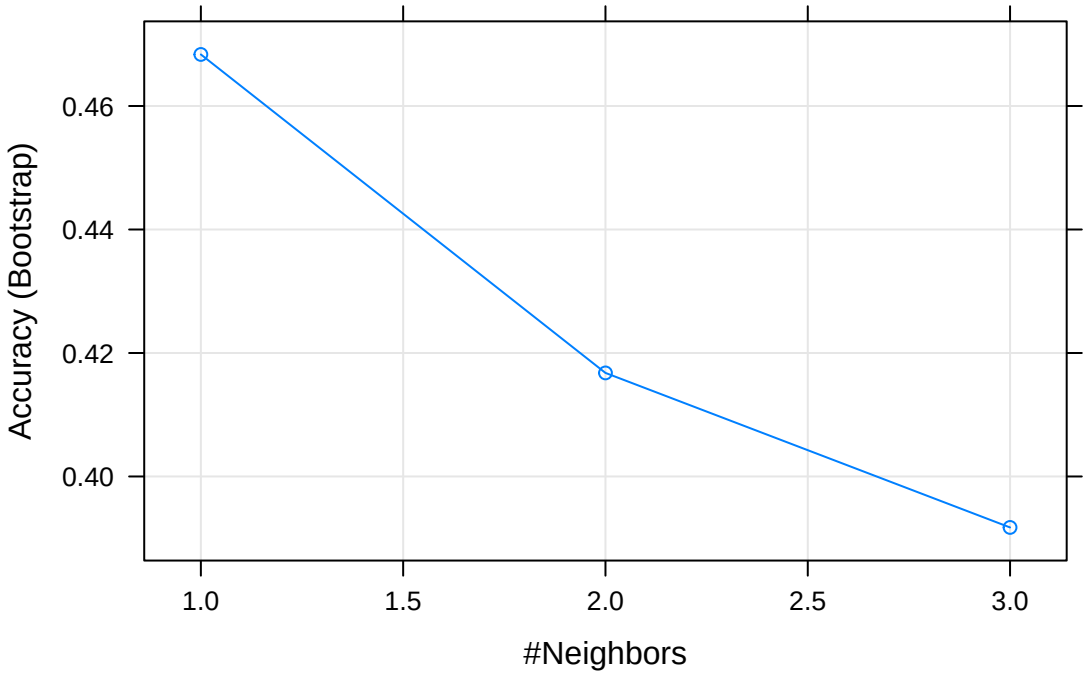
The accuracy obtained with this method, computed on the test set, is: 0.907

The computational time required on a 2.4Ghz processor is: 57 minutes

3.7 Knn

To try a different type of model we implement a K Nearest Neighbors predictor, using **Caret** again.
We obtain an accuracy of 0.516. As we can see the performance of this method for our problem is quite low.

We also notice that the best value of k (number of neighbors) chosen by the tuning process is k=1, and the accuracy monotonically decreases with higher values of k:



The reason of this poor performance may be that, as we saw in the principal component analysis, the most part of the row to row variability of our data is not related to our target prediction variable, so distance based methods are not well suited for this problem.

The computational time required on a 2.4Ghz processor is: 30 seconds

3.8 Flexible Discriminant analysis

In the data exploration section we noticed that a lot of our predictors are almost normally distributed. From the plot showing the numbers of rows related to each of our Severity outcomes we also understand that a successful method should be able to account for prevalence.
We thus decide to use Flexible Discriminant Analysis, this method in an extensions of Linear Discriminant Analysis that uses non-linear combinations of predictors.
We can implement this model with the **Caret**, **earth** and **mda** packages:

```
fit_fda <- train(Severity ~ ., mtehod = "fda", data = train_set_dummy)
prediction_fda <- predict(fit_fda, test_set_dummy)
confMatrix_fdat <- confusionMatrix(prediction_fda, reference = test_set_dummy$Severity)
```

When assessing the accuracy of this model on the test set we obtain a very good result: 0.954 .

The computational time required on a 2.4Ghz processor is: 9 minutes

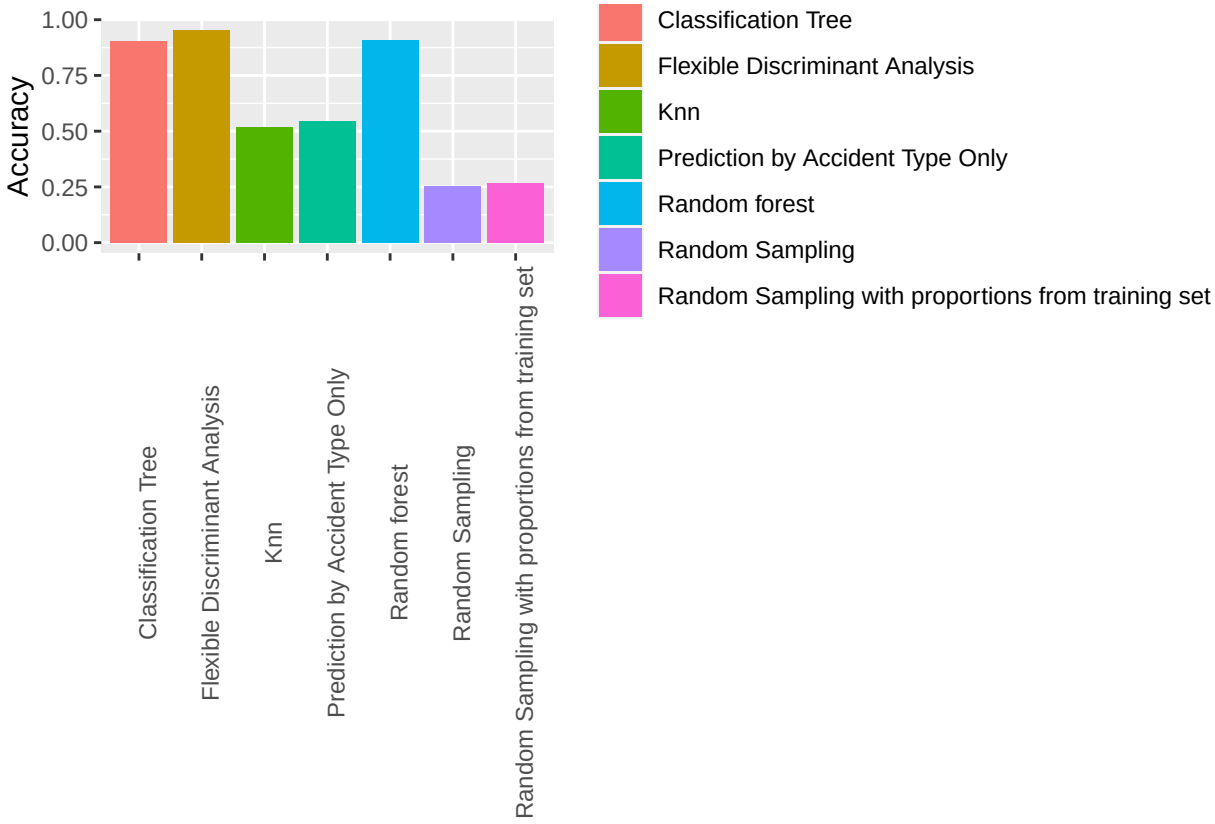
4 Results

Since our prediction model is not intended to entirely substitute a human, because it does not perform all the necessary steps in the report classification process, our goal is to provide the Safety Investigator with an algorithm that requires the lowest possible number of manual interventions in the part of the process delegated to the computer.
To achieve this goal our algorithm has to reach the highest number of correct predictions possible.
We will thus focus on accuracy as our performance variable.

During the previous chapter we saw that distance based models do not obtain good results, decision tree based models instead greatly increase the performance of our predictions and discriminant analysis models help us in further improving our accuracy.

As far as we are concerned with computational time we see that in this case, when overfitting in Classification Trees does not give problems, the RandomForest method requires much more time without obtaining an accuracy increase. Using flexible discriminant analysis instead we trade a longer (but still acceptable) computational time for a better accuracy.

We compare the result of the various methods implemented in the plot below:



5 Conclusion

Even if during data explorations no clear correlation seemed to be present between our predictors and our target prediction variable, machine learning methods based on decision trees and discriminant analysis managed to obtain an acceptable accuracy for an application aimed at helping the human operator, without the need to completely take over his decisions.

Unfortunately the data found publicly available on keggle does not perfectly mimic the data usually transmitted with aviation safety reports, so further work is needed to build another algorithm on the actual data managed by an airline/airport's Safety Management System.
Practical implementations also have to take into account that the predictors available may vary from company to company depending on the safety reporting system used.
Another interesting topic is the linking between the safety reporting system and the flight data monitoring system (an application that downloads data recorded by the airplane flight data recorder after every flight), machine learning applications able to mix data recorded by sensor and data reported by humans would be a great step forward on this matter.