NEOs Classification

Alberto Sinigaglia, Beatrice Sofia Bertipaglia, Chiara Colato, Flavia Gianfrate

20/7/2022

Project Presentation

In this project we face a *binary classification* problem of celestial bodies called *NEOs*. These are objects belonging to the Solar System whose orbit can intersect the Earth's orbit and therefore can represent a danger for our planet. NEOs are therefore classified into "Hazardous" and "Not Hazardous", based on parameters that take into account their potential approach to the Earth, represented by the attributes of the proposed dataset. The aim of this analysis is therefore to define whether these celestial bodies can be dangerous or not for the Earth and to understand which characteristics are useful for this goal.

Dataset Presentation and Preprocessing

The proposed dataset consists of 4687 objects whose characteristics are described by 40 attributes. These are orbital parameters with mostly real values, suitable for describing the orbit of the body and its structure. The variable that describes the danger of the object is presented, on the other hand, as a logical variable and it's called *Hazardous*. We proceed with the importation of the data and necessary libraries.

```r
# Retrieval of the libraries necessary for the correct import of
# the data, manipulation and data visualization

library(tidyverse)
library(ggplot2)
library(gridExtra)
library(tidymodels)
library(leaps)
library(glmnet)
library(pROC)
library(rsample)
library(correlation)
library(DataExplorer)
library(knitr)
library(corrplot)
library(regclass)
library(rsample)
library(corrplot)
library(outliers)
library(dplyr)
library(caret)
library(class)
```

Data Uploading

```r
# We upload the dataset "nasa.csv" with the function "read_delim" of
# the tidyverse package:
```

```r
nasa_orig <- read_delim("nasa.csv", delim = ",")
```

```
## Rows: 4687 Columns: 40
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (2): Orbiting Body, Equinox
## dbl  (35): Neo Reference ID, Name, Absolute Magnitude, Est Dia in KM(min), E...
## lgl   (1): Hazardous
## dttm  (1): Orbit Determination Date
## date  (1): Close Approach Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# We check if the upload is correct looking at the first 5 rows

head(nasa_orig,5)
```

```
## # A tibble: 5 x 40
##   `Neo Reference ID`    Name `Absolute Magni~` `Est Dia in KM~` `Est Dia in KM~`
##                <dbl>   <dbl>             <dbl>            <dbl>            <dbl>
## 1            3703080 3703080              21.6          0.127             0.284
## 2            3723955 3723955              21.3          0.146             0.327
## 3            2446862 2446862              20.3          0.232             0.518
## 4            3092506 3092506              27.4          0.00880           0.0197
## 5            3514799 3514799              21.6          0.127             0.284
## # ... with 35 more variables: `Est Dia in M(min)` <dbl>,
## #   `Est Dia in M(max)` <dbl>, `Est Dia in Miles(min)` <dbl>,
## #   `Est Dia in Miles(max)` <dbl>, `Est Dia in Feet(min)` <dbl>,
## #   `Est Dia in Feet(max)` <dbl>, `Close Approach Date` <date>,
## #   `Epoch Date Close Approach` <dbl>, `Relative Velocity km per sec` <dbl>,
## #   `Relative Velocity km per hr` <dbl>, `Miles per hour` <dbl>,
## #   `Miss Dist.(Astronomical)` <dbl>, `Miss Dist.(lunar)` <dbl>, ...
```

```r
# We rename the columns' names in order to able to recall them: the
# presence of spaces, parenthesis or points could be a problem, so we
# avoid the issues introducing instead the character "_"

names(nasa_orig)<- gsub("\\s","_",names(nasa_orig))
```

Data Pre-Processing

Before proceeding with the analysis we explore the structure of the dataset we want to study.

The first operation we do is to check the presence of missing values in the dataset: if the output returned is *FALSE* we don't have any missing value, so we can proceed.

```r
anyNA(nasa_orig)
```

```
## [1] FALSE
```

We consider now the features described in the dataset that represent the characteristics for each unit. We searched for the meaning of each one and we report the descriptions in the following table:

Variable	Description
Neo Reference ID	ID of the object
Name	Object name
Absolute Magnitude	Absolute magnitude is a measure of the intrinsic brightness of an object that does not consider its brightness variations due to real conditions
Est Dia in KM(min)	Minimum estimated diameter in km
Est Dia in KM(max)	Maximum estimated diameter in km
Est Dia in M(min)	Minimum estimated diameter in m
Est Dia in M(max)	Maximum estimated diameter in m
Est Dia in Miles(min)	Minimum estimated diameter in Miles
Est Dia in Miles(max)	Maximum estimated diameter in Miles
Est Dia in Feet(min)	Minimum estimated diameter in Feet
Est Dia in Feet(max)	Maximum estimated diameter in Feet
Close Approach Date	Date where the object is nearer to Earth
Epoch Date Close Approach	Coordinates where the object is nearer to Earth
Relative Velocity km per sec	Velocity in km/s
Relative Velocity km per hr	Velocity in km/hr
Miles per hour	Velocity in Miles/hr
Miss Dist.(Astronomical)	Distance from Earth where the object passes in AU
Miss Dist.(lunar)	Distance from Earth where the object passes in LU
Miss Dist.(kilometers)	Distance from Earth where the object passes in km
Miss Dist.(miles)	Distance from Earth where the object passes in Miles
Orbiting Body	Body around which the object orbits
Orbit ID	Orbit ID of the object
Orbit Determination Date	Date when the orbit was defined
Orbit Uncertainity	Uncertainity measure of the orbit, related to several parameters used in the orbit determination process (as the number of measurements, the time spanned by those observations, their quality and the geometry of the observations
Minimum Orbit Intersection	Minimum distance between object's orbit and Earth orbit
Jupiter Tisserand Invariant	Tisserand parameter (or Tisserand invariant), is a value calculated from several orbital elements (semi-major axis, eccentricity and orbital inclination) of a relatively small object. It is used to distinguish different types of orbits
Epoch Osculation	Time to which the data refer
Eccentricity	The eccentricity can be considered as the measure of how far the orbit is deviated from a circle
Semi Major Axis	Semi-major axis length in AU
Inclination	Inclination is one of the orbital parameters that describe the shape and orientation of an orbit: it's the angular distance of the orbital plane from the reference plane expressed in degrees
Asc Node Longitude	The ascending node is the point where the orbit of the object passes through the plane of reference
Orbital Period	The orbital period is the amount of time a given astronomical object takes to complete one orbit around another object
Perihelion Distance	Maximum distance of the object from the Earth
Perihelion Arg	Parametrically, it is the angle from the ascending node of the body measured in the direction of movement
Aphelion Dist	Minimum distance of the object from the Earth
Perihelion Time	Moment when the object is in perihelion
Mean Anomaly	In celestial mechanics, the mean anomaly is the fraction of an elliptical orbit period that has elapsed since the orbiting body passed periapsis
Mean Motion	Mean motion is used as an approximation of the actual orbital speed in making an initial calculation of the body's position in its orbit
Equinox	Time to which the data refer

Variable	Description
Hazardous	Dangerousness of the object - response variable

Starting from the set of the original features, some considerations about them need to be done:

- Some variables are redundant, as they are presented for different units of measure;
- Some features are simple identifiers;
- Some features show a unique value for all units and therefore do not bring useful information;

In order to avoid to consider the same information several times, we proceed removing the repeated features, maintaining only one unit measure: for example we choose to consider the measurement of the *Estimated Diameter* in Kilometers (instead of Miles or Feet). Then, we delete also the identifiers *Name* and *ID* because they are not useful for our analysis.

```
toremove<- c("Neo_Reference_ID","Name", "Est_Dia_in_M(min)",
             "Est_Dia_in_M(max)", "Est_Dia_in_Miles(min)",
             "Est_Dia_in_Miles(max)", "Est_Dia_in_Feet(min)",
             "Est_Dia_in_Feet(max)", "Close_Approach_Date",
             "Epoch_Date_Close_Approach",
             "Relative_Velocity_km_per_sec",
             "Miles_per_hour", "Miss_Dist.(lunar)",
             "Miss_Dist.(kilometers)",
             "Miss_Dist.(miles)", "Orbiting_Body", "Orbit_ID",
             "Orbit_Determination_Date", "Epoch_Osculation",
             "Equinox", "Est_Dia_in_KM(min)")

nasa<- nasa_orig %>%
       select(-all_of(toremove))

colnames(nasa)[2]<- "Est_Dia_in_KM_max"
colnames(nasa)[4]<- "Miss_Dist_Astronomical"
```

Another thing that we have to check is the type of our features. We look at their nature and what we would like is to codify them in a correct and meaningful way in such a way that we can use them inside models.

```
# Conversion in correct type:

summary(nasa)
```

```
##  Absolute_Magnitude Est_Dia_in_KM_max  Relative_Velocity_km_per_hr
##  Min.   :11.16      Min.   : 0.00226   Min.   :  1208
##  1st Qu.:20.10      1st Qu.: 0.07482   1st Qu.: 30358
##  Median :21.90      Median : 0.24777   Median : 46504
##  Mean   :22.27      Mean   : 0.45751   Mean   : 50295
##  3rd Qu.:24.50      3rd Qu.: 0.56760   3rd Qu.: 65080
##  Max.   :32.10      Max.   :34.83694   Max.   :160681
##  Miss_Dist_Astronomical Orbit_Uncertainity Minimum_Orbit_Intersection
##  Min.   :0.0001779      Min.   :0.000      Min.   :0.0000021
##  1st Qu.:0.1334196      1st Qu.:0.000      1st Qu.:0.0145851
##  Median :0.2650286      Median :3.000      Median :0.0473655
##  Mean   :0.2567782      Mean   :3.517      Mean   :0.0823201
##  3rd Qu.:0.3841541      3rd Qu.:6.000      3rd Qu.:0.1235935
##  Max.   :0.4998841      Max.   :9.000      Max.   :0.4778910
##  Jupiter_Tisserand_Invariant  Eccentricity      Semi_Major_Axis
##  Min.   :2.196                Min.   :0.007522   Min.   :0.6159
```

```
##  1st Qu.:4.050              1st Qu.:0.240858   1st Qu.:1.0006
##  Median :5.071              Median :0.372450   Median :1.2410
##  Mean   :5.056              Mean   :0.382569   Mean   :1.4003
##  3rd Qu.:6.019              3rd Qu.:0.512411   3rd Qu.:1.6784
##  Max.   :9.025              Max.   :0.960261   Max.   :5.0720
##   Inclination       Asc_Node_Longitude Orbital_Period   Perihelion_Distance
##  Min.   : 0.01451   Min.   :  0.0019   Min.   : 176.6   Min.   :0.08074
##  1st Qu.: 4.96234   1st Qu.: 83.0812   1st Qu.: 365.6   1st Qu.:0.63083
##  Median :10.31184   Median :172.6254   Median : 504.9   Median :0.83315
##  Mean   :13.37384   Mean   :172.1573   Mean   : 635.6   Mean   :0.81338
##  3rd Qu.:19.51168   3rd Qu.:255.0269   3rd Qu.: 794.2   3rd Qu.:0.99723
##  Max.   :75.40667   Max.   :359.9059   Max.   :4172.2   Max.   :1.29983
##  Perihelion_Arg     Aphelion_Dist     Perihelion_Time    Mean_Anomaly
##  Min.   :  0.0069   Min.   :0.8038   Min.   :2450100   Min.   :  0.0032
##  1st Qu.: 95.6259   1st Qu.:1.2661   1st Qu.:2457815   1st Qu.: 87.0069
##  Median :189.7616   Median :1.6182   Median :2457973   Median :185.7189
##  Mean   :183.9322   Mean   :1.9871   Mean   :2457728   Mean   :181.1679
##  3rd Qu.:271.7776   3rd Qu.:2.4512   3rd Qu.:2458108   3rd Qu.:276.5319
##  Max.   :359.9931   Max.   :8.9839   Max.   :2458839   Max.   :359.9180
##   Mean_Motion      Hazardous
##  Min.   :0.08628   Mode :logical
##  1st Qu.:0.45329   FALSE:3932
##  Median :0.71295   TRUE :755
##  Mean   :0.73824
##  3rd Qu.:0.98467
##  Max.   :2.03900
```

```
nasa$Orbit_Uncertainity<- nasa$Orbit_Uncertainity %>%
                          as.factor() %>%
                          as.numeric()

nasa$Perihelion_Distance<- nasa$Perihelion_Distance %>%
                           as.factor() %>%
                           as.numeric()

nasa <- nasa %>% mutate(Hazardous = ifelse(Hazardous == TRUE, 1, 0))

nasa$Hazardous<- nasa$Hazardous %>%
                           as.factor()

# Our response variable is codified now as a factor:
# Hazardous = 0 means that the unit is not considered dangerous for Earth
# Hazardous = 1 means that the unit is considered dangerous for Earth
```

Data Balance Check

From the summary another issue emerges: the classes *Hazardous* and *Not Hazardous* are not balanced and this could become a problem during the training of the classification models.

```
prop.table(table(nasa$Hazardous))
```

```
##
##         0         1
## 0.8389162 0.1610838
```

In order to limit the problem we decide to introduce techniques able to balance the two classes, sampling more from the minority class and less from the majority one. In our analysis, after the subdivision of the dataset, we will consider the possibility to balance the training dataset: for the first models we will compare their performances on the original dataset and on the balanced one, in order to understand if the balancing can bring to an improvements of the performances.

Splitting in Training e Test Sets

Before continuing to explore the properties of our data, we divide them into two different datasets: the *training dataset* and the *test dataset* that will be used respectively for training our models and for testing the models' performances. We proceed with all the considerations about the features focusing on the training one, trying to understand the relations between them. The test set will remain "unknown". For doing the split, the proportions defined are 0.75 and 0.25, respectively for the two sets.

```
# Train-Test Split

# We set a fixed seed in order to be able to reproduce the same
# results each time we run the code:
set.seed(0607)

split <- initial_split(nasa, prop = 0.75)
train <- training(split)
test <- testing(split)

# We check if the proportion of the Hazardous NEOs is the same in all the datasets:
# we want to have datasets that represent the original situation.

# Proportion of Hazardous and Not Hazardous in the training dataset:

prop.table(table(train$Hazardous))
```

```
##
##         0         1
## 0.8421053 0.1578947
```

```
# Proportion of Hazardous and Not Hazardous in the test dataset:

prop.table(table(test$Hazardous))
```

```
##
##         0         1
## 0.8293515 0.1706485
```

Data Analysis

We focus now on the training dataset in order to explore the relations between the different features and the relation between the features and the response variable. This procedure can be useful for evaluating from an objective point of view the connections between the available characteristics we have and for finding a solution to our problem.

Correlations

The first step of our data analysis consists into the analysis of *Correlation*, statistical measure used to quantify the strength of the linear relationship between two variables and compute their association. We are basically trying to observe the level of change in one variable due to the change in the other one. Although we are computing only the linear connection between the covariates, this can be a useful starting point. First, we can look at the correlations between our response variable *Hazardous* and the covariates. At the beginning we consider *Hazardous* as a numeric variable in order to have a path to follow in this correlation analysis, but ,because in our problem *Hazardous* is a factor, then we proceed plotting the density of the most correlated features, separating the case for bodies classified as *Hazardous* and *Not Hazardous*.

Variable	Correlation with Hazardous
Absolute Magnitude	-0.32
Orbit_Uncertainity	-0.32
Minimum Orbit Intersection	-0.29
Perihelion Distance	-0.21
Eccentricity	0.19
Relative Velocity	0.18

```r
Haz<- train$Hazardous

# Absolute Magnitude
a<- ggplot(train, aes(x = Absolute_Magnitude, fill = Haz)) +
  geom_density(alpha = 0.4)+
  ggtitle("Absolute Magnitude - Density Plot") +
  xlab("Absolute Magnitude")

# Orbit Uncertainity

b<- ggplot(train, aes(x = Orbit_Uncertainity, fill = Haz)) +
  geom_bar(aes(y = ..prop..),position = "dodge") +
  ggtitle("Orbit Uncertainity - Bar Plot") +
  xlab("Orbit Uncertainity")

# Minimum Orbit Intersection
c<- ggplot(train, aes(x = Minimum_Orbit_Intersection, fill = Haz)) +
  geom_density(alpha = 0.4) +
  xlim(0, 0.5) +
  ggtitle("Minimum Orbit Intersection - Density Plot") +
  xlab("Minimum Orbit Intersection")

# Eccentricity
d<- ggplot(train, aes(x = Eccentricity, fill = Haz)) +
  geom_density(alpha = 0.4)+
  ggtitle("Eccentricity - Density Plot") +
  xlab("Eccentricity")

# Perihelion Distance
e<- ggplot(train, aes(x = Perihelion_Distance, fill = Haz)) +
  geom_density(alpha = 0.4) +
  ggtitle("Perihelion Distance - Density Plot") +
  xlab("Perihelion Distance")

# Relative Velocity
f<- ggplot(train, aes(x = Relative_Velocity_km_per_hr, fill = Haz)) +
  geom_density(alpha = 0.4) +
  ggtitle("Relative Velocity - Density Plot") +
  xlab("Relative Velocity")
```

```r
grid.arrange(a, b, c, d, e, f, nrow = 2)
```

Figure 1: Density plots for NEOs hazardous and not hazardous

As we note, we see that the values of *Absolute Magnitude* distribute differently for NEOs dangerous and not dangerous for Earth; the same seems to hold for *Minimum Orbit Intersection* and *Orbit Uncertainity*. Looking at the *Eccentricity* plot, we see that the mean of the distribution changes as the level of *Hazardous* changes, and, considering that we are looking at the range (0,1), translations of the mean from 0.25 to 0.5 can be considered relevant differences. Proceeding with the plots we can observe also that, generally, for bodies which are classified as potentially dangerous we have less values of *Perihelion Distance*. The *Relative Velocity*, instead, doesn't seem to differentiate well the two classes of objects.

These initial information leads to the identification of a dangerous NEO as a body with a medium-low absolute magnitude, with a degree of Uncertainity of the orbit and minimum distance of intersection with the Earth orbit generally low. We can also say that, usually, these are bodies that show a relative speed a little greater than a body not considered dangerous and that generally show a shorter perihelion distance.

Focusing now on the correlations between the covariates, we can build the following matrix:

```r
# We use the "corrplot" function that represents the correlation in a graphic way:

corrplot(cor(train[,-19]),
         method = "number",
         diag = FALSE,
         tl.cex = 0.4,
         number.cex = 0.5,
         tl.col = "black")
```

We note that we have some variables that are highly correlated between each other. This can represent a case of *Collinearity* between the attributes: we try to understand the nature of these relationships studying deeper the meaning of the variables mostly involved.

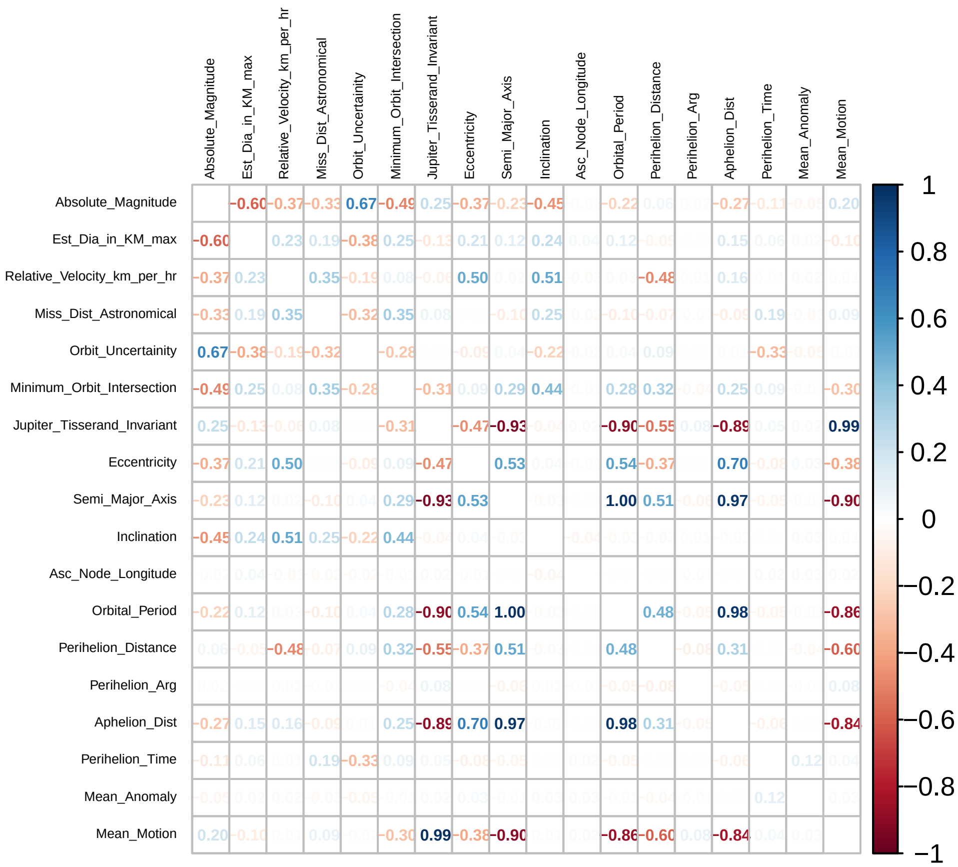


Figure 2: Correlation matrix between the covariates

Partial Correlations

Thanks to the *Partial Correlation* we can investigate the relationship between two variables without the influence of any other relationship between them and other attributes present in the dataset.

```
# We use the "correlation" function from the "correlation" package, where we can specify
# the computation of the partial correlations

correlation(train, partial = TRUE)

# (The output is not shown for its dimensions, but we resume the highest
# values into the following table).
```

We report in the following table the highest individuated correlations:

Variable 1	Variable 2	Partial Correlation
Mean Motion	Jupiter Tisserand Parameter	0.99
Mean Motion	Semi Major Axis	-0.9
Mean Motion	Orbital Period	-0.86
Mean Motion	Aphelion Distance	-0.84
Aphelion Distance	Jupiter Tisserand Parameter	-0.89
Aphelion Distance	Mean Motion	0.84
Aphelion Distance	Eccentricity	0.7
Aphelion Distance	Semi Major Axis	0.97
Aphelion Distance	Orbital Period	0.98
Perihelion Distance	Jupiter Tisserand Parameter	-0.55
Perihelion Distance	Mean Motion	-0.6
Perihelion Distance	Semi Major Axis	0.51
Orbital Period	Jupiter Tisserand Parameter	-0.9
Orbital Period	Semi Major Axis	1
Inclination	Relative Velocity	0.51
Semi Major Axis	Jupiter Tisserand Parameter	-0.93
Semi Major Axis	Eccentricity	0.53
Eccentricity	Relative Velocity	0.5
Orbit Uncertainity	Absolute Magnitude	0.67
Estimated Diameter KM max	Absolute Magnitude	-0.6

In order to understand why such variables are so correlated, we investigate more on their formulations.

- First we analyze the relations where is involved the value of *Jupiter Tisserand Parameter*. In particular we have that the formula for computing it contains the information about the *Semi Major Axis* a, the *Eccentricity* e and the *Inclination* i of the orbit:

$$T_p = \frac{a_p}{a} + 2cos(i)\sqrt{\frac{a}{a_p}(1 - e^2)}$$

Knowing it, we can explain clearly why there's the presence of a so high correlation between such an invariant and the mentioned attributes.

- Then we analyze the relation between *Mean Motion* and the *Orbital Period*: the mean motion is simply computed dividing one revolution for the orbital period. This explains their negative correlation.
- We consider now the *Orbital Period* quantity. In its formulation is considered the length of the *Semi Major Axis* and for that reason we have a so high correlation between them.

- We note also the relation between *Absolute Magnitude* and the *Estimated Diameter*. In order to better define their relationship we can observe the formula with which they are connected:

$$D = \frac{1329}{\sqrt{p}} 10^{-0.2H}$$

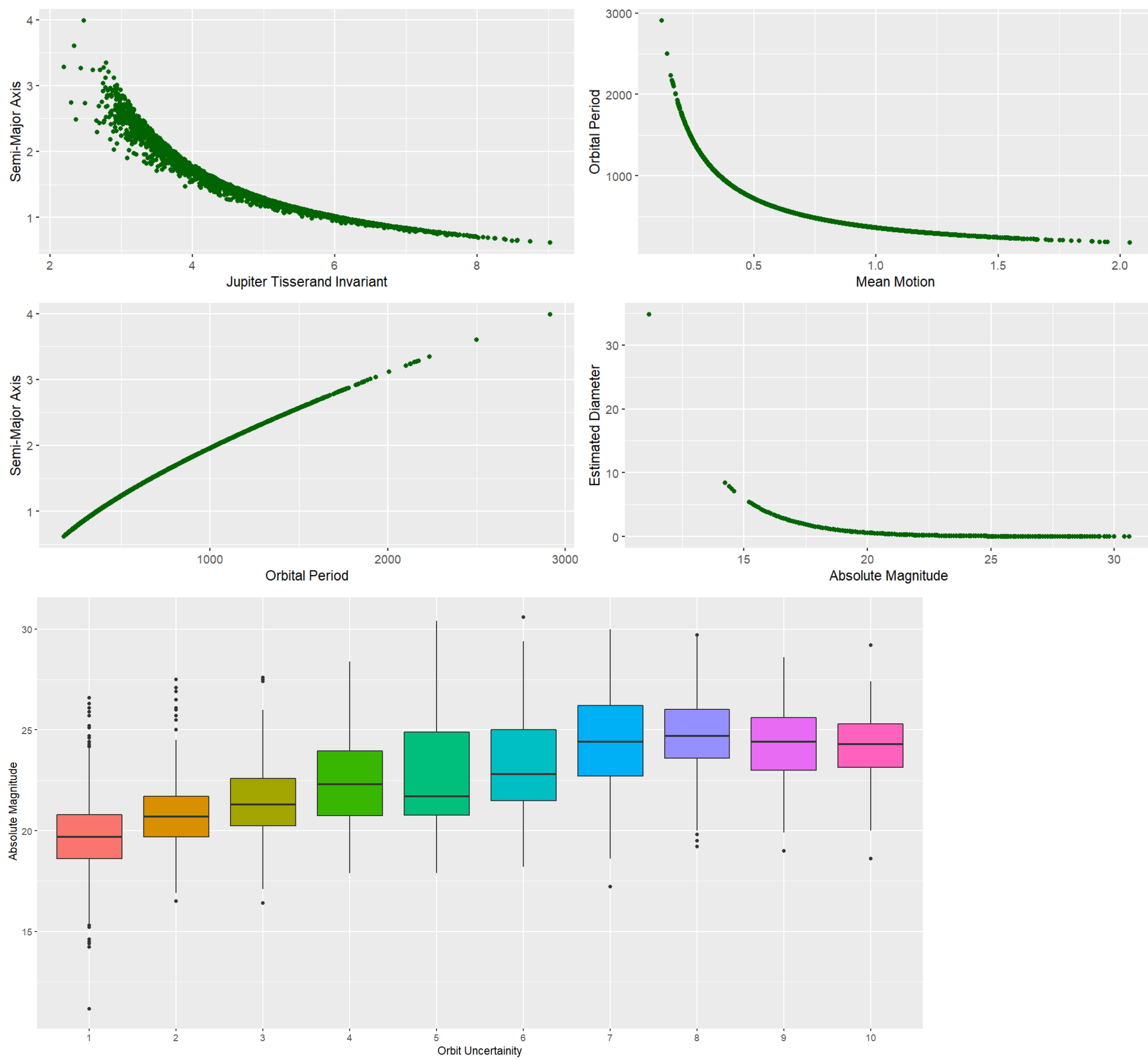
where p corresponds to the *albedo* and H represents the *Absolute Magnitude*. We can see, in fact, that they are inversely proportional, and so we can explain the negative correlation.

- Then we have also that the value of the *Orbit Uncertainity* is positively correlated with the *Absolute Magnitude* of a body. We know that the smaller the magnitude value, the brighter the body, so can be reasonable to think that to grater values of magnitude correspond darker bodies for which is difficult to define perfectly their orbit trajectory.

These relations can be clearly represented by the following plots:

```
a1<- ggplot(data = train, aes(Jupiter_Tisserand_Invariant, Semi_Major_Axis)) +
  geom_jitter(color = "dark green") +
  xlab("Jupiter Tisserand Invariant") +
  ylab("Semi-Major Axis")

a2<- ggplot(data = train, aes(Mean_Motion, Orbital_Period)) +
  geom_jitter(color = "dark green") +
  xlab("Mean Motion") +
  ylab("Orbital Period")

a3<- ggplot(data = train, aes(Orbital_Period, Semi_Major_Axis)) +
  geom_jitter(color = "dark green")  +
  xlab("Orbital Period") +
  ylab("Semi-Major Axis")

a4<- ggplot(data = train, aes(Absolute_Magnitude, Est_Dia_in_KM_max))+
  geom_jitter(color = "dark green") +
  xlab("Absolute Magnitude") +
  ylab("Estimated Diameter")

a5<- ggplot(data = train, aes(as.factor(Orbit_Uncertainity), Absolute_Magnitude,
                              fill = as.factor(Orbit_Uncertainity))) +
  geom_boxplot() +
  labs(x = "Orbit Uncertainity", y = " Absolute Magnitude") +
  theme(legend.position = "none")


grid.arrange(a1, a2, a3, a4, nrow = 2)
```

```
a5
```



All this existent connections, as we resume in the Figure 3, between the covariates could become a problem during the definition of the classification models. For each case, in fact, we will discuss how to face the presence of such relationships and how to treat them.

Logistic Regression Models Definition

We start our analysis considering the *Logistic Regression Model* for classifying the NEOs into *Hazardous* and *Not Hazardous* classes. In particular we try to build our model on the original training dataset and on the training dataset that we balance, in order to have at most the same quantities of dangerous and not dangerous example. We also consider the possibility to apply a *Stepwise Selection* approach for including and excluding iteratively the covariates inside the model. Then, for each built structure we will consider different thresholds for the classification: we are in fact interested into minimize the quantity of *False Negatives*. In order to create a good model useful for the identification of the potentially hazardous bodies, we prefer, in fact, to have a less values for *Hazardous* NEOs classified as *Not Hazardous*,maintaining at the same time not too high the quantity of the *False Positives*.

```
library(ROSE)
library(caret)
```

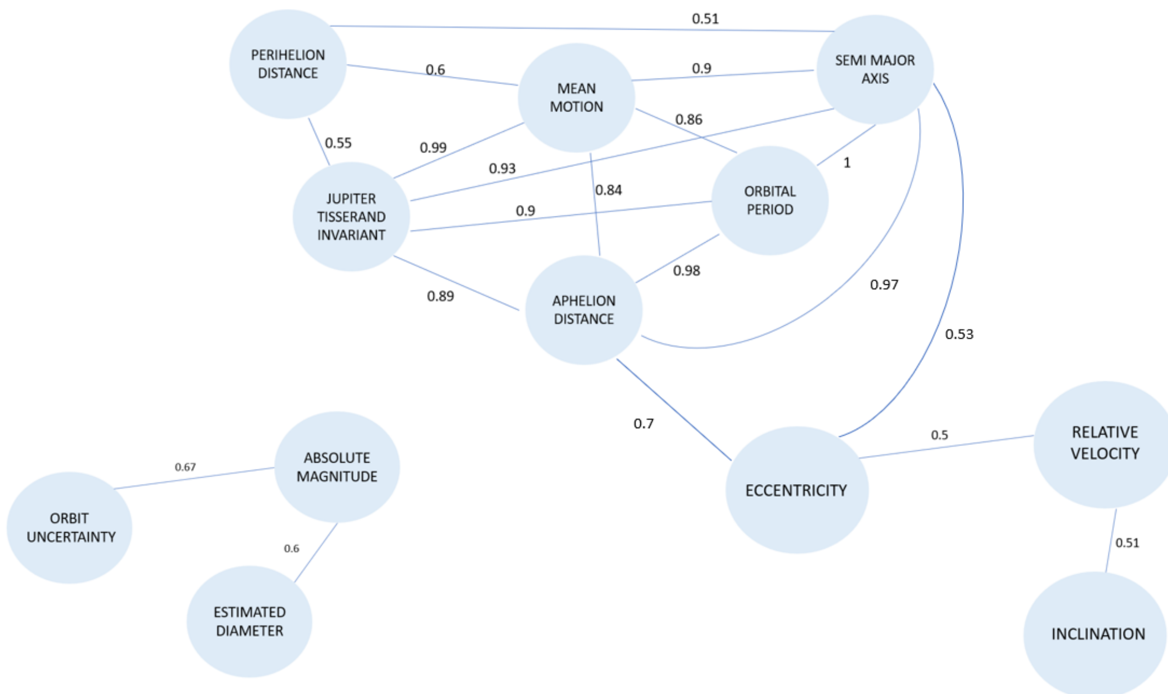


Figure 3: Relations Map for Features

```r
library(cowplot)
library(leaps)

# We use the function "ovun.sample" from the "ROSE" package that
# creates possibly balanced samples by random over-sampling
# minority examples and under-sampling majority examples.

train_balanced<- ovun.sample(Hazardous~., data = train,
                             method = "both", p = 0.5,
                             N = 3515, seed = 1)$data


# Calling the "data" part of the output we obtain the resulting
# new dataset.


# We define three different thresholds for the classification task:
# 0.4, 0.5, 0.6

threshold4<- 0.4
threshold5<- 0.5
threshold6<- 0.6
```

Simple Logistic Model

With logistic regression we can estimate the probability of an event happening based of the values of the other variables: here we are estimating the probability of a body to be *Hazardous*, given the values of its attributes.

After an initial definition of the model, we check the presence of collinearity using the *VIF* function that calculates the *Variance Inflation Factor* of all predictors. The VIF of a predictor is a measure for how easily it is predicted from a linear regression using the other predictors. In general, a VIF larger than 1/(1-R2), where R2 is the Multiple R-squared of the regression, indicates that predictor is more related to the other predictors than it is to the response.

Considering the computed quantity and removing a feature at time, we reach at the end a situation where we have two variables with high VIF value : *Absolute Magnitude* and *Est_Dia_in_KM_max*. We should delete *Absolute Magnitude* because of its highest value, but we observe that, if we decide for its removal the total errors produced in the classification becomes twice as much the total error of the model that includes the same attribute. For this reason we consider the possibility of eliminating *Est_Dia_in_KM_max* instead of *Absolute Magnitude* and see what happens. In this way, we kept *Absolute Magnitude* in our model which is now less than 1/(1-R2) and we achieved good results in terms of total error of the model, eliminating the collinearity between the variables of our dataset. Hence all the features considered in the model carry information regarding their relationship to the answer, without considering spurious correlations.

```r
# Model definition:

glm_compl<- glm(data = train,
            Hazardous ~ .,
            family = "binomial")

# We compute the reference level R-Squared

s<- summary(glm_compl)
r2<- 1 - (s$deviance/s$null.deviance)

1/(1-r2)
```

```
## [1] 4.512277
```

```r
# Using the VIF function and comparing the obtained values with the
# computed quantity:
# (The process is done iteratively where we delete one variable at time)

VIF(glm_compl)
```

```
##          Absolute_Magnitude            Est_Dia_in_KM_max
##                   11.030324                     6.229586
## Relative_Velocity_km_per_hr       Miss_Dist_Astronomical
##                    2.722461                     1.232429
##           Orbit_Uncertainity  Minimum_Orbit_Intersection
##                    1.647146                     3.028814
## Jupiter_Tisserand_Invariant                 Eccentricity
##                 2731.476027                    34.073306
##              Semi_Major_Axis                  Inclination
##                 2213.590294                     6.716405
##           Asc_Node_Longitude               Orbital_Period
##                    1.042881                  2586.020995
##          Perihelion_Distance               Perihelion_Arg
##                   43.415305                     1.030216
##                Aphelion_Dist              Perihelion_Time
##                 3307.158240                     1.165570
##                 Mean_Anomaly                  Mean_Motion
##                    1.056344                  1621.811958
```

```r
glm_compl<- glm(data = train,
            Hazardous ~.-Aphelion_Dist-Semi_Major_Axis-
              Jupiter_Tisserand_Invariant-
              Eccentricity-Mean_Motion-Est_Dia_in_KM_max,
            family = "binomial")

# Observation of the model summary:

summary(glm_compl)
```

```
##
## Call:
## glm(formula = Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis -
##     Jupiter_Tisserand_Invariant - Eccentricity - Mean_Motion -
##     Est_Dia_in_KM_max, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7114  -0.1252  -0.0222   0.0000   2.6850
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 3.580e+02  2.412e+02   1.485   0.1376
## Absolute_Magnitude         -1.249e+00  7.909e-02 -15.789  < 2e-16 ***
## Relative_Velocity_km_per_hr -4.480e-07  4.783e-06  -0.094   0.9254
## Miss_Dist_Astronomical      5.508e-03  6.353e-01   0.009   0.9931
## Orbit_Uncertainity         -2.626e-01  4.153e-02  -6.325 2.53e-10 ***
## Minimum_Orbit_Intersection -1.114e+02  6.181e+00 -18.017  < 2e-16 ***
## Inclination                 2.162e-02  1.082e-02   1.999   0.0456 *
## Asc_Node_Longitude         -9.204e-04  8.488e-04  -1.084   0.2782
## Orbital_Period              6.768e-04  3.442e-04   1.966   0.0493 *
## Perihelion_Distance        -4.547e-05  1.396e-04  -0.326   0.7446
## Perihelion_Arg             -7.488e-05  8.639e-04  -0.087   0.9309
## Perihelion_Time            -1.335e-04  9.814e-05  -1.361   0.1736
## Mean_Anomaly                7.291e-04  8.291e-04   0.879   0.3792
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3066.22  on 3514  degrees of freedom
## Residual deviance:  902.83  on 3502  degrees of freedom
## AIC: 928.83
##
## Number of Fisher Scoring iterations: 9
```

```r
# Computing the predictions with the model on the test set:

pred_glm_compl<- predict(glm_compl, test, type = "response")

# Converting the prediction in {0,1} according to the chosen threshold:

pred_glm_compl_04<- ifelse(pred_glm_compl > threshold4, 1, 0)
pred_glm_compl_05<- ifelse(pred_glm_compl > threshold5, 1, 0)
```

```r
pred_glm_compl_06<- ifelse(pred_glm_compl > threshold6, 1, 0)
```

We can observe the resulting summary of the model. First we have the explicitation of the formula where it's possible to see what variables we are including or excluding in the model. Then we move our attention on the estimated coefficient for each of them: the significant attributes seem to be:

- Absolute Magnitude
- Orbit Uncertainity
- Minimum Orbit Intersection
- Inclination
- Orbital Period

In order to know how good the logistic model is classifying we can take a look to the *Confusion Matrix* built for each threshold:

```r
# Confusion matrix with threshold = 0.4

table(test$Hazardous, pred_glm_compl_04)
mean(pred_glm_compl_04!=test$Hazardous)

# Confusion matrix with threshold = 0.5

table(test$Hazardous, pred_glm_compl_05)
mean(pred_glm_compl_05!=test$Hazardous)

# Confusion matrix with threshold = 0.6

table(test$Hazardous, pred_glm_compl_06)
mean(pred_glm_compl_06!=test$Hazardous)
```

Confusion Matrices - Simple GLM

Treshold: 0.4

	Predictions		
Real Values	0	1	Total
0	935	37	972
1	27	173	200
Total	962	210	1172

Treshold: 0.5

	Predictions		
Real Values	0	1	Total
0	946	26	972
1	35	165	200
Total	981	191	1172

Treshold: 0.6

	Predictions		
Real Values	0	1	Total
0	952	20	972
1	48	152	200
Total	1000	172	1172

Focusing now on the confusion matrices, we note that increasing the threshold for which we classify a body as *Hazardous*, we risk to produce more *False Negatives*. Instead decreasing the same value we could be able to obtain a model that can individuate the potentially dangers and so it could be more useful for avoiding them.

Logistic Model with Stepwise Selection

We introduce here, to the previous defined model, the *Stepwise Selection* that consists into a method that iteratively adds and/or removes predictors. The aim is to find the subset of variables in the dataset that results in the best performing model, that is a model that minimizes the AIC quantity. The *Akaike Information Criterion* (AIC) is an estimator of prediction error and of the relative quality of statistical models for a given set of data. In particular, the main characteristic of this method is the introduction of a *penalization term*, with which the model is penalized as much variables includes. This term, fore some models, including the logistic regression, is computed in a closed form.

The Stepwise can be implemented following three strategies:

- *Forward selection*: it starts with no predictors in the model and it adds the most contributive predictors in an iterative way, stopping when the improvement is no significant.
- *Backward selection*: it starts with all predictors in the model (full model) and iteratively removes the least contributive predictors, stopping when you we a model where all predictors are statistically

significant.

- We can also combine both the techniques: we start with no predictors, then sequentially add the most contributive predictors (like forward selection). After adding each new variable, remove any variables that no longer provide an improvement in the model fit (like backward selection).

In our project we use the third option:

```
library(leaps)
library(MASS)
```

```
##
## Caricamento pacchetto: 'MASS'

## Il seguente oggetto è mascherato da 'package:dplyr':
##
##     select
```

```
# Model definition:

# Here we don't re-apply the VIF method because we start from the
# previous result.

glm_compl<- glm(data = train,
            Hazardous ~.-Aphelion_Dist-Semi_Major_Axis-
              Jupiter_Tisserand_Invariant-
              Eccentricity-Mean_Motion-Est_Dia_in_KM_max,
            family = "binomial")

# Application of the Stepwise method, specifying that we consider
# both the forward and the backward directions. We consider as
# reference metric the Akaike Information Criterion:

glm_compl_step <- stepAIC(glm_compl, direction = "both",
                          trace = FALSE)

# Observation of the model summary:

summary(glm_compl_step)
```

```
##
## Call:
## glm(formula = Hazardous ~ Absolute_Magnitude + Orbit_Uncertainity +
##     Minimum_Orbit_Intersection + Inclination + Orbital_Period,
##     family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.6614  -0.1266  -0.0225   0.0000   2.8362
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 2.991e+01  1.784e+00  16.765  < 2e-16 ***
## Absolute_Magnitude         -1.259e+00  7.609e-02 -16.545  < 2e-16 ***
## Orbit_Uncertainity         -2.460e-01  3.908e-02  -6.295 3.08e-10 ***
## Minimum_Orbit_Intersection -1.109e+02  6.097e+00 -18.195  < 2e-16 ***
## Inclination                 2.081e-02  9.191e-03   2.264   0.0235 *
```

```
## Orbital_Period              6.226e-04  2.720e-04   2.289   0.0221 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3066.22  on 3514  degrees of freedom
## Residual deviance:  906.85  on 3509  degrees of freedom
## AIC: 918.85
##
## Number of Fisher Scoring iterations: 9
```

```
# Computing the predictions with the model on the test set:

pred_glm_compl_step = predict(glm_compl_step, test, type = "response")


# Converting the predictions in {0,1} according to the chosen threshold:

pred_glm_compl_step_04 = ifelse(pred_glm_compl_step > threshold4, 1, 0)
pred_glm_compl_step_05 = ifelse(pred_glm_compl_step > threshold5, 1, 0)
pred_glm_compl_step_06 = ifelse(pred_glm_compl_step > threshold6, 1, 0)
```

Also here we take a look to the summary of the model. Here the significant variables are:

- Absolute Magnitude
- Orbit Uncertainity
- Minimum Orbit Intersection
- Inclination
- Orbital Period

As we can note, comparing this result with the previous one, the model considers as significant the same features as before. The only thing that changes is the significance of the intercept.

```
# Confusion matrix with threshold = 0.4

table(test$Hazardous, pred_glm_compl_step_04)
mean(pred_glm_compl_step_04!=test$Hazardous)

# Confusion matrix with threshold = 0.5

table(test$Hazardous, pred_glm_compl_step_05)
mean(pred_glm_compl_step_05!=test$Hazardous)

# Confusion matrix with threshold = 0.6

table(test$Hazardous, pred_glm_compl_step_06)
mean(pred_glm_compl_step_06!=test$Hazardous)
```

Confusion Matrices - GLM with the Stepwise approach

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	934	38	972
1	25	175	200
Total	959	213	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	945	27	972
1	35	165	200
Total	980	192	1172

Treshold: 0.6			
	Predictions		
Real Values	0	1	Total
0	952	20	972
1	46	154	200
Total	998	174	1172

We compare now the *Confusion Matrices* obtained with the simple complete model and with the model that uses the stepwise. What we observe is that basically the results are the same: the difference are very minimum. We can deduce that the use of the *Stepwise* doesn't bring to better results.

Logistic Model with Balancing

Over-sampling and Under-sampling In the presence of an unbalanced distribution of the response variable, the learning process can be distorted: the model tends, in fact, to focus on the majority class and ignore rare events. The solution adopted in this project is based on *over-sampling* and *under-sampling*: it consists on a pre-treatment of the data, which has the advantage of being independent of any classification model and adaptable to many different contexts. The goal is to modify the distribution of the classes so as to alleviate the degree of imbalance. This process could not be useful for every proposed model, so what we will do is to consider for each model its performance deriving from the training on the original set of data and from the balanced one.

```r
# Model definition:

glm_bal<- glm(data = train_balanced,
              Hazardous ~ .,
              family = "binomial")

# We compute the reference level R-Squared

s<- summary(glm_bal)
r2<- 1 - (s$deviance/s$null.deviance)

1/(1-r2)
```

```
## [1] 5.210651
```

```r
# Using the VIF function and comparing the obtained values with the
# computed quantity:
# (The process is done iteratively where we delete one variable at time)

VIF(glm_bal)
```

```
##           Absolute_Magnitude            Est_Dia_in_KM_max
##                     8.081116                     4.584000
## Relative_Velocity_km_per_hr      Miss_Dist_Astronomical
##                     3.069135                     1.434706
##          Orbit_Uncertainity  Minimum_Orbit_Intersection
##                     1.752583                     2.689621
## Jupiter_Tisserand_Invariant                Eccentricity
##                  2746.299632                    35.917461
##              Semi_Major_Axis                 Inclination
##                  2631.052128                     6.912107
##           Asc_Node_Longitude              Orbital_Period
##                     1.072416                 3059.090309
##          Perihelion_Distance              Perihelion_Arg
##                    43.591526                     1.043203
##               Aphelion_Dist             Perihelion_Time
##                  3274.311712                     1.201623
##                 Mean_Anomaly                 Mean_Motion
##                     1.111289                 1627.167844
```

```r
glm_bal<- glm(data = train_balanced,
              Hazardous ~.-Aphelion_Dist-Semi_Major_Axis-
```

```r
              Jupiter_Tisserand_Invariant-Eccentricity-
                Est_Dia_in_KM_max-Mean_Motion,
              family = "binomial")

# Observation of the model summary:

summary(glm_bal)
```

```
##
## Call:
## glm(formula = Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis -
##     Jupiter_Tisserand_Invariant - Eccentricity - Est_Dia_in_KM_max -
##     Mean_Motion, family = "binomial", data = train_balanced)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.3604  -0.0872   0.0000   0.1851   2.1521
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  1.472e+02  2.307e+02   0.638  0.52354
## Absolute_Magnitude          -1.412e+00  7.665e-02 -18.421  < 2e-16 ***
## Relative_Velocity_km_per_hr -4.177e-06  4.177e-06  -1.000  0.31737
## Miss_Dist_Astronomical      -8.597e-01  5.950e-01  -1.445  0.14850
## Orbit_Uncertainity          -2.884e-01  3.629e-02  -7.947 1.91e-15 ***
## Minimum_Orbit_Intersection  -1.160e+02  5.500e+00 -21.091  < 2e-16 ***
## Inclination                  1.457e-02  9.815e-03   1.485  0.13762
## Asc_Node_Longitude          -1.015e-03  7.620e-04  -1.332  0.18281
## Orbital_Period               8.833e-04  3.068e-04   2.879  0.00398 **
## Perihelion_Distance         -1.823e-04  1.204e-04  -1.514  0.12993
## Perihelion_Arg              -1.306e-03  7.495e-04  -1.742  0.08142 .
## Perihelion_Time             -4.522e-05  9.393e-05  -0.481  0.63020
## Mean_Anomaly                 7.907e-04  7.210e-04   1.097  0.27275
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4869.9  on 3514  degrees of freedom
## Residual deviance: 1191.0  on 3502  degrees of freedom
## AIC: 1217
##
## Number of Fisher Scoring iterations: 8
```

```r
# Computing the predictions with the model on the test set:

pred_glm_bal<- predict(glm_bal, test, type = "response")


# Converting the predictions in {0,1} according to the chosen threshold:

pred_glm_bal_04<- ifelse(pred_glm_bal > threshold4, 1, 0)
pred_glm_bal_05<- ifelse(pred_glm_bal > threshold5, 1, 0)
pred_glm_bal_06<- ifelse(pred_glm_bal > threshold6, 1, 0)
```

Comparing the resulting significant variables between the *Simple GLM* and the same model applied on the balanced training dataset, we can see immediately note that we have the presence, again of the same significant variables:

- Absolute Magnitude
- Minimum Orbit Intersection
- Orbit Uncertainity
- Orbital Period

```r
# Confusion matrix with threshold = 0.4

table(test$Hazardous, pred_glm_bal_04)
mean(pred_glm_bal_04!=test$Hazardous)

# Confusion matrix with threshold = 0.5

table(test$Hazardous, pred_glm_bal_05)
mean(pred_glm_bal_05!=test$Hazardous)

# Confusion matrix with threshold = 0.6

table(test$Hazardous, pred_glm_bal_06)
mean(pred_glm_bal_06!=test$Hazardous)
```

Confusion Matrices - GLM with Balanced dataset

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	868	104	972
1	7	193	200
Total	875	297	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	890	82	972
1	11	189	200
Total	901	271	1172

Treshold: 0.6			
	Predictions		
Real Values	0	1	Total
0	908	64	972
1	15	185	200
Total	923	249	1172

We can suddenly notice that the total error is increased. In particular we were able to reduce the *False Negative* rate, as we wanted, but, in the while we increased the *False Positive* rate. Although it misclassifies a greater number of observations, this model could be considered "safer" and so preferable for our aim of finding dangerous bodies for the Earth.

Balancing with weights We can propose another method for balancing the classes: it consists into add different weights to elements belonging to different classes and in particular we want to valorize examples of the minority one. For that reason, we apply a weight greater than 1 to the *Hazardous* class.

```r
# Definition of the weights

w<- rep(1, nrow(train))

sum(train$Hazardous==0)/sum(train$Hazardous==1)
```

```
## [1] 5.333333
```

```r
w[train$Hazardous == 1]<- 5

# Model definition:

glm_weighted<- glm(data = train,
               Hazardous ~ .,
               family = "binomial", weights = w)

# We compute the reference level R-Squared
```

```r
s<- summary(glm_weighted)
r2<- 1 - (s$deviance/s$null.deviance)

1/(1-r2)
```

```
## [1] 5.21696
```

```r
# Using the VIF function and comparing the obtained values with the
# computed quantity:
# (The process is done iteratively where we delete one variable at time)

VIF(glm_weighted)
```

```
##          Absolute_Magnitude           Est_Dia_in_KM_max
##                    7.527514                    4.145426
## Relative_Velocity_km_per_hr      Miss_Dist_Astronomical
##                    2.847903                    1.339717
##          Orbit_Uncertainity  Minimum_Orbit_Intersection
##                    1.694125                    2.679198
## Jupiter_Tisserand_Invariant                Eccentricity
##                 2748.417110                   35.867567
##             Semi_Major_Axis                 Inclination
##                 2719.193092                    6.108253
##          Asc_Node_Longitude              Orbital_Period
##                    1.055791                 3210.808278
##         Perihelion_Distance              Perihelion_Arg
##                   43.953641                    1.036599
##               Aphelion_Dist             Perihelion_Time
##                 3428.951857                    1.186606
##                Mean_Anomaly                 Mean_Motion
##                    1.075292                 1596.844214
```

```r
glm_weighted<- glm(data = train,
              Hazardous ~.-Aphelion_Dist-Semi_Major_Axis-
              Jupiter_Tisserand_Invariant-Eccentricity-
              Mean_Motion-Est_Dia_in_KM_max,
              family = "binomial", weights = w)

# Observation of the model summary:

summary(glm_weighted)
```

```
##
## Call:
## glm(formula = Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis -
##     Jupiter_Tisserand_Invariant - Eccentricity - Mean_Motion -
##     Est_Dia_in_KM_max, family = "binomial", data = train, weights = w)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.3874  -0.2330  -0.0390  -0.0001   4.4133
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  2.973e+02  1.624e+02   1.831 0.067042 .
```

```
## Absolute_Magnitude          -1.447e+00  6.122e-02 -23.631  < 2e-16 ***
## Relative_Velocity_km_per_hr -2.585e-06  3.366e-06  -0.768 0.442615
## Miss_Dist_Astronomical      -2.524e-01  4.523e-01  -0.558 0.576819
## Orbit_Uncertainity          -2.459e-01  2.840e-02  -8.661  < 2e-16 ***
## Minimum_Orbit_Intersection  -1.148e+02  4.299e+00 -26.697  < 2e-16 ***
## Inclination                  1.713e-02  7.810e-03   2.194 0.028240 *
## Asc_Node_Longitude          -1.561e-03  5.788e-04  -2.697 0.006998 **
## Orbital_Period               8.225e-04  2.378e-04   3.458 0.000544 ***
## Perihelion_Distance         -6.413e-05  9.535e-05  -0.673 0.501237
## Perihelion_Arg               7.520e-05  5.913e-04   0.127 0.898804
## Perihelion_Time             -1.064e-04  6.608e-05  -1.610 0.107502
## Mean_Anomaly                 1.169e-03  5.617e-04   2.081 0.037417 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7944.4  on 3514  degrees of freedom
## Residual deviance: 1932.3  on 3502  degrees of freedom
## AIC: 1958.3
##
## Number of Fisher Scoring iterations: 8
```

```r
# Computing the predictions with the model on the test set:

pred_glm_weighted<- predict(glm_weighted, test, type = "response")

# Converting the predictions in {0,1} according to the chosen threshold:

pred_glm_weighted_04<- ifelse(pred_glm_weighted > threshold4, 1, 0)
pred_glm_weighted_05<- ifelse(pred_glm_weighted > threshold5, 1, 0)
pred_glm_weighted_06<- ifelse(pred_glm_weighted > threshold6, 1, 0)
```

As we can see from the summary above, we note that here we have the introduction of two different variables, that are: *Missing Distance* and *Inclination*, there isn't instead the measure of the *Orbital Period*

- Absolute Magnitude
- Missing Distance
- Orbit Uncertainity
- Minimum Orbit Intersection
- Inclination

We look now at the confusion matrices in order to understand if this new model could be a good alternative to the two previous proposed:

```r
# Confusion matrix with threshold = 0.4

table(test$Hazardous, pred_glm_weighted_04)
mean(pred_glm_weighted_04!=test$Hazardous)

# Confusion matrix with threshold = 0.5

table(test$Hazardous, pred_glm_weighted_05)
mean(pred_glm_weighted_05!=test$Hazardous)

# Confusion matrix with threshold = 0.6
```

```
table(test$Hazardous, pred_glm_weighted_06)
mean(pred_glm_weighted_06!=test$Hazardous)
```

Confusion Matrices - GLM with weighted classes

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	869	103	972
1	6	194	200
Total	875	297	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	890	82	972
1	11	189	200
Total	901	271	1172

Treshold: 0.6			
	Predictions		
Real Values	0	1	Total
0	909	63	972
1	14	186	200
Total	923	249	1172

From the obtained results we can see that the method that we use for balancing the dataset doesn't change the results in terms of goodness of classification of our models. Then, we have the possibility to obtain at most the same results using less variables. For that, we choose to continue to consider only the over-sampling and under-sampling technique.

Logistic Model with Balancing and Stepsize

To conclude with the Logistic Regression models we try to combine the two ideas: we develop a model using the *Stepwise* across the variables selected starting from the balanced dataset.

```
# Application of the Stepwise method :

# (Here we don't re-apply the VIF method because we start from the
# previous result.)

#We start from the glm_bal model where we have already applied the
# VIF process, then we proceed with the Stepwise method.

glm_bal_step<- stepAIC(glm_bal, direction = "both",
                       trace = FALSE)

# Observing the summary of the model:

summary(glm_bal_step)
```

```
##
## Call:
## glm(formula = Hazardous ~ Absolute_Magnitude + Miss_Dist_Astronomical +
##     Orbit_Uncertainity + Minimum_Orbit_Intersection + Orbital_Period +
##     Perihelion_Arg, family = "binomial", data = train_balanced)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.4540  -0.0861   0.0000   0.1931   2.2146
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                3.667e+01  1.742e+00  21.048  < 2e-16 ***
## Absolute_Magnitude        -1.452e+00  7.224e-02 -20.105  < 2e-16 ***
## Miss_Dist_Astronomical    -1.007e+00  5.370e-01  -1.874   0.0609 .
## Orbit_Uncertainity        -2.677e-01  3.287e-02  -8.143 3.84e-16 ***
## Minimum_Orbit_Intersection -1.156e+02  5.408e+00 -21.374  < 2e-16 ***
## Orbital_Period             5.673e-04  2.390e-04   2.373   0.0176 *
## Perihelion_Arg            -1.151e-03  7.424e-04  -1.551   0.1210
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4869.9  on 3514  degrees of freedom
## Residual deviance: 1197.5  on 3508  degrees of freedom
## AIC: 1211.5
##
## Number of Fisher Scoring iterations: 8
```

```r
# Computing the predictions with the model on the test set:

pred_glm_bal_step<- predict(glm_bal_step, test, type = "response")


# Converting the predictions in {0,1} according to the chosen threshold:

pred_glm_bal_step_04 = ifelse(pred_glm_bal_step > threshold4, 1, 0)
pred_glm_bal_step_05 = ifelse(pred_glm_bal_step > threshold5, 1, 0)
pred_glm_bal_step_06 = ifelse(pred_glm_bal_step > threshold6, 1, 0)
```

The significant variables now are:

- Absolute Magnitude
- Minimum Orbit Intersection
- Orbit Uncertainity
- Orbital Period

As we can see, the maintained covariates are the same of the model with only tha balancing of the classes. The Stepwise approach also in this case doesn't contribute to change the structure of the model.

```r
# Confusion matrix with threshold = 0.4

table(test$Hazardous, pred_glm_bal_step_04)
mean(pred_glm_bal_step_04!=test$Hazardous)

# Confusion matrix with threshold = 0.5

table(test$Hazardous, pred_glm_bal_step_05)
mean(pred_glm_bal_step_05!=test$Hazardous)

# Confusion matrix with threshold = 0.6

table(test$Hazardous, pred_glm_bal_step_06)
mean(pred_glm_bal_step_06!=test$Hazardous)
```

Confusion Matrices - GLM with Balanced dataset and Stepwise approach

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	868	104	972
1	7	193	200
Total	875	297	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	888	84	972
1	13	187	200
Total	901	271	1172

Treshold: 0.6			
	Predictions		
Real Values	0	1	Total
0	909	63	972
1	16	184	200
Total	925	247	1172

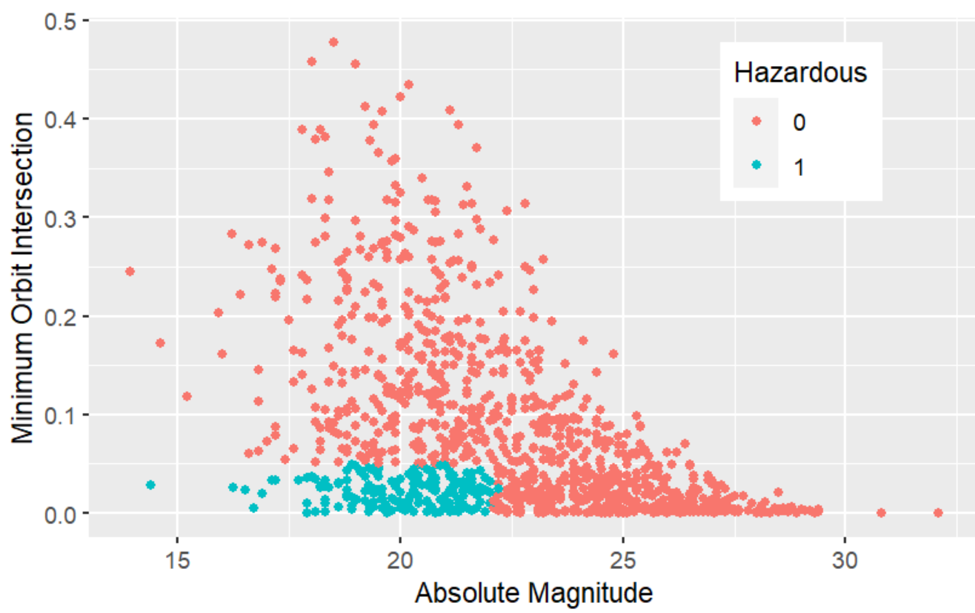
Comparing the results obtained with the balanced dataset without and with the stepwise, we can see that the difference is minimum: in particular they have the same variables, so the contribute of the step is considerable irrelevant.

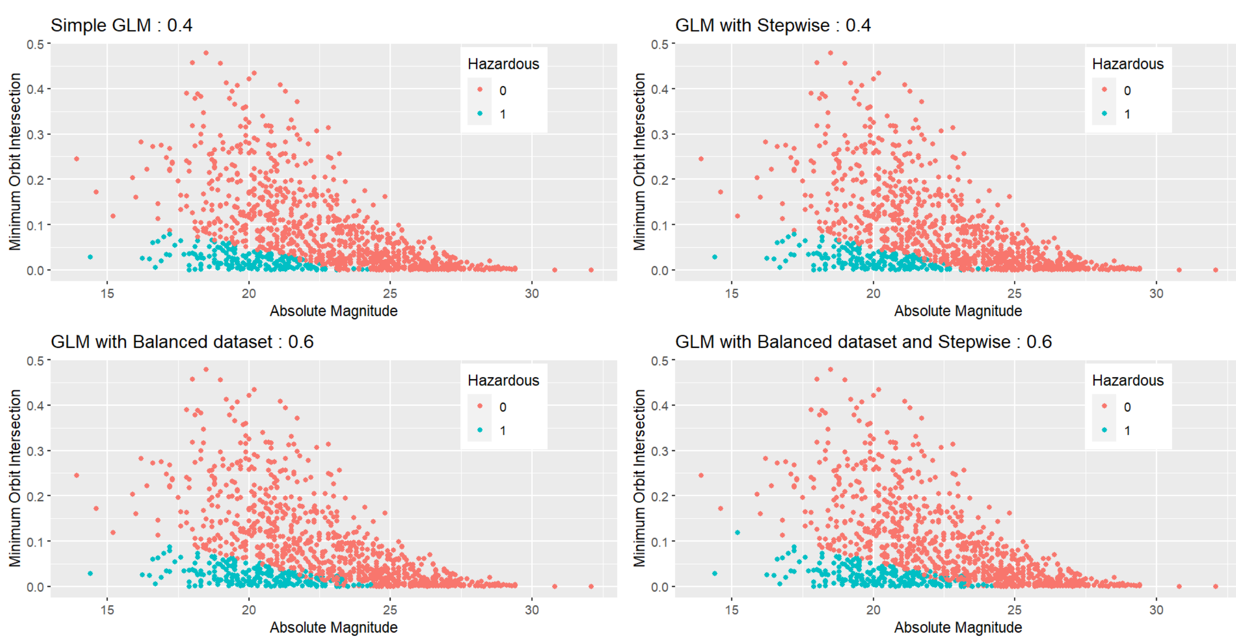
We can compare the ability of classification of the models, looking at how they are able to reproduce the original graphical classification of the points that we can see in the following plots.

```
Haz_test<- test$Hazardous

# First we present the original classification :

ggplot(test, aes(x = Absolute_Magnitude,
                 y = Minimum_Orbit_Intersection,
                 color = Haz_test)) +
  geom_point()+
  labs(x = "Absolute Magnitude",
       y = "Minimum Orbit Intersection",
       color = "Hazardous")  +
  theme(legend.position = c(0.8, 0.8))
```

```
# The we try to reproduce the same plot as above, considering the
# classifications obtained with the models

a<- ggplot(test, aes(x = Absolute_Magnitude,
                 y = Minimum_Orbit_Intersection,
                 color = as.factor(pred_glm_compl_04))) +
  geom_point()+
  labs(x = "Absolute Magnitude",
       y = "Minimum Orbit Intersection",
       color = "Hazardous",
       title = "Simple GLM : 0.4")  +
  theme(legend.position = c(0.8, 0.8))



b<- ggplot(test, aes(x = Absolute_Magnitude,
                 y = Minimum_Orbit_Intersection,
                 color = as.factor(pred_glm_compl_step_04))) +
  geom_point()+
  labs(x = "Absolute Magnitude",
       y = "Minimum Orbit Intersection",
       color = "Hazardous",
       title = "GLM with Stepwise : 0.4")  +
  theme(legend.position = c(0.8, 0.8))



c<- ggplot(test, aes(x = Absolute_Magnitude,
                 y = Minimum_Orbit_Intersection,
                 color = as.factor(pred_glm_bal_06))) +
  geom_point()+
  labs(x = "Absolute Magnitude",
       y = "Minimum Orbit Intersection",
       color = "Hazardous",
       title = "GLM with Balanced dataset : 0.6")  +
  theme(legend.position = c(0.8, 0.8))


d<- ggplot(test, aes(x = Absolute_Magnitude,
```

```
                    y = Minimum_Orbit_Intersection,
                    color = as.factor(pred_glm_bal_step_06))) +
  geom_point()+
  labs(x = "Absolute Magnitude",
       y = "Minimum Orbit Intersection",
       color = "Hazardous",
       title = "GLM with Balanced dataset and Stepwise : 0.6")  +
  theme(legend.position = c(0.8, 0.8))


grid.arrange(a, b, c, d, nrow = 2)
```

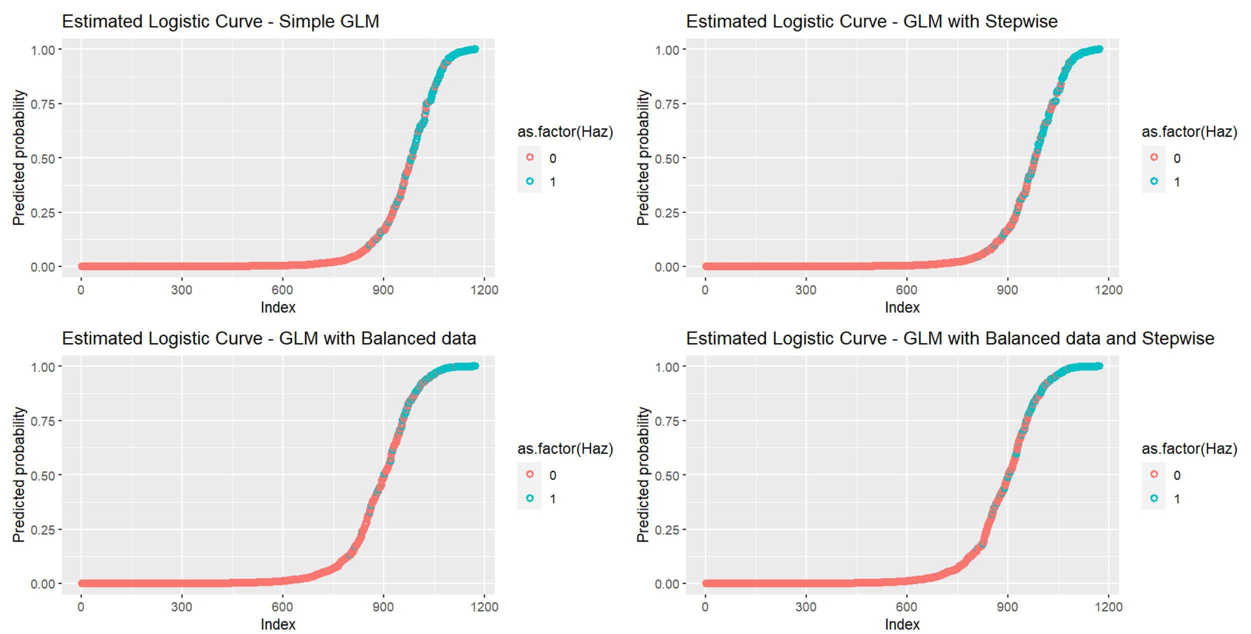




As we said looking at the confusion matrices computed for the different models, we can see also graphically that the main difference is visible between models trained on the original dataset and the ones trained on the balanced set of data. In particular here we can also see that the *Stepwise* approach doesn't help to produce better classifications. Focusing on the bottom figures, we can see that, comparing the results with the real differentiation of bodies, we classify as *Hazardous* more observations that in reality are *Not Hazardous*, but at the same moment we are able to classify almost all the dangerous units as they are. In any case, the difference between all the *GLM* models is not so relevant.

```
# We compare the results obtained with the four different models, plotting
# now an estimation of the logistic curve using the predictions given by
# the models:

predicted_data<- data.frame(prob.of.Haz = pred_glm_compl, Haz = test$Hazardous)
predicted_data<- predicted_data[order(predicted_data$prob.of.Haz, decreasing = FALSE),]
predicted_data$rank<-  1:nrow(predicted_data)

a<- ggplot(data = predicted_data, aes(x = rank, y = prob.of.Haz)) +
  geom_point(aes(color = as.factor(Haz)), alpha = 1, shape = 1, stroke = 1) +
  xlab("Index")+
  ylab("Predicted probability")+
  ggtitle("Estimated Logistic Curve - Simple GLM")


predicted_data<- data.frame(prob.of.Haz = pred_glm_compl_step, Haz = test$Hazardous)
predicted_data<- predicted_data[order(predicted_data$prob.of.Haz, decreasing = FALSE),]
predicted_data$rank<- 1:nrow(predicted_data)

b<- ggplot(data = predicted_data, aes(x = rank, y = prob.of.Haz)) +
  geom_point(aes(color = as.factor(Haz)), alpha = 1, shape = 1, stroke = 1) +
  xlab("Index")+
  ylab("Predicted probability")+
  ggtitle("Estimated Logistic Curve - GLM with Stepwise")



predicted_data<- data.frame(prob.of.Haz = pred_glm_bal, Haz = test$Hazardous)
predicted_data<- predicted_data[order(predicted_data$prob.of.Haz, decreasing = FALSE),]
predicted_data$rank<- 1:nrow(predicted_data)

c<- ggplot(data = predicted_data, aes(x = rank, y = prob.of.Haz)) +
  geom_point(aes(color = as.factor(Haz)), alpha = 1, shape = 1, stroke = 1) +
  xlab("Index")+
  ylab("Predicted probability")+
  ggtitle("Estimated Logistic Curve - GLM with Balanced data")



predicted_data<- data.frame(prob.of.Haz = pred_glm_bal_step, Haz = test$Hazardous)
predicted_data<- predicted_data[order(predicted_data$prob.of.Haz, decreasing = FALSE),]
predicted_data$rank<- 1:nrow(predicted_data)

d<- ggplot(data = predicted_data, aes(x = rank, y = prob.of.Haz)) +
  geom_point(aes(color = as.factor(Haz)), alpha = 1, shape = 1, stroke = 1) +
  xlab("Index")+
  ylab("Predicted probability")+
```

```r
  ggtitle("Estimated Logistic Curve - GLM with Balanced data and Stepwise")


library(gridExtra)
grid.arrange(a, b, c, d, nrow = 2)
```



In the plot we put in the *x-axis* our observations sorted considering their probability to be dangerous, in the *y-axis* we have instead the predicted probability. We observe that in all models, highest levels of probability bring all the model to classify the observations related as *Hazardous*. In the middle of the curves we find those values trivial to classify, from which we can see the different performances of the proposed models. Observing the colors proportions, we see that with the balanced dataset used for the training we are able to respect more the correct proportion of *Hazardous* bodies present in the test set. In fact, from the graph and from the previous confusion matrix, we can see that the balanced dataset is able to classify the celestial bodies labeled Hazardous much better also because it has many more observations of that class to train on.

Discriminant Analysis

Besides the generalized linear models, other useful models for our classification problem could be *Linear Discriminant Analysis* and *Quadratic Discriminant Analysis* which are based on Bayes theorem and try different approach for classification. Before being able to apply them, it is necessary to check some assumptions underlying these methodologies.

The most important, that we check before starting with the modelling, is the normality of the variables conditioned to the two classes *Hazardous* and *Not Hazardous*.

Normality Requirement of the covariates

We introduce at this point the *Shapiro-Wilk test* that's the most powerful test for verifying the normality. It's used for the verification of statistic hypothesis, where in particular the null hypothesis is represented by the normality of the population distribution; if and only if the value of p is equal to or less than 0.05, then the hypothesis of normality will be rejected by the Shapiro test. On failing, the test can state that the data will not fit the distribution normally with 95% confidence.

```r
# We apply the Shapiro - Wilks test on each covariate, considering
# the two different classes:
```

```r
shapiro.test(train_balanced$Absolute_Magnitude[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Absolute_Magnitude[train_balanced$Hazardous == 0]
## W = 0.99036, p-value = 1.398e-09
```

```r
shapiro.test(train_balanced$Absolute_Magnitude[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Absolute_Magnitude[train_balanced$Hazardous == 1]
## W = 0.94646, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Est_Dia_in_KM_max[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Est_Dia_in_KM_max[train_balanced$Hazardous == 0]
## W = 0.23046, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Est_Dia_in_KM_max[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Est_Dia_in_KM_max[train_balanced$Hazardous == 1]
## W = 0.64943, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Relative_Velocity_km_per_hr[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Relative_Velocity_km_per_hr[train_balanced$Hazardous == 0]
## W = 0.94053, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Relative_Velocity_km_per_hr[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Relative_Velocity_km_per_hr[train_balanced$Hazardous == 1]
## W = 0.94559, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Miss_Dist_Astronomical[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Miss_Dist_Astronomical[train_balanced$Hazardous == 0]
## W = 0.95186, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Miss_Dist_Astronomical[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Miss_Dist_Astronomical[train_balanced$Hazardous == 1]
## W = 0.94835, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Orbit_Uncertainity[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Orbit_Uncertainity[train_balanced$Hazardous == 0]
## W = 0.86877, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Orbit_Uncertainity[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Orbit_Uncertainity[train_balanced$Hazardous == 1]
## W = 0.66425, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Minimum_Orbit_Intersection[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Minimum_Orbit_Intersection[train_balanced$Hazardous == 0]
## W = 0.86683, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Minimum_Orbit_Intersection[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Minimum_Orbit_Intersection[train_balanced$Hazardous == 1]
## W = 0.96019, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Jupiter_Tisserand_Invariant[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Jupiter_Tisserand_Invariant[train_balanced$Hazardous == 0]
## W = 0.97717, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Jupiter_Tisserand_Invariant[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Jupiter_Tisserand_Invariant[train_balanced$Hazardous == 1]
## W = 0.97937, p-value = 5.77e-15
```

```
shapiro.test(train_balanced$Eccentricity[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  train_balanced$Eccentricity[train_balanced$Hazardous == 0]
## W = 0.98871, p-value = 1.136e-10
```

```
shapiro.test(train_balanced$Eccentricity[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Eccentricity[train_balanced$Hazardous == 1]
## W = 0.98927, p-value = 6.507e-10
```

```
shapiro.test(train_balanced$Semi_Major_Axis[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Semi_Major_Axis[train_balanced$Hazardous == 0]
## W = 0.9116, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Semi_Major_Axis[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Semi_Major_Axis[train_balanced$Hazardous == 1]
## W = 0.91477, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Inclination[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Inclination[train_balanced$Hazardous == 0]
## W = 0.88741, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Inclination[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Inclination[train_balanced$Hazardous == 1]
## W = 0.87563, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Asc_Node_Longitude[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Asc_Node_Longitude[train_balanced$Hazardous == 0]
## W = 0.96079, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Asc_Node_Longitude[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Asc_Node_Longitude[train_balanced$Hazardous == 1]
## W = 0.95783, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Orbital_Period[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Orbital_Period[train_balanced$Hazardous == 0]
## W = 0.87065, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Orbital_Period[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Orbital_Period[train_balanced$Hazardous == 1]
## W = 0.86743, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Perihelion_Distance[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Perihelion_Distance[train_balanced$Hazardous == 0]
## W = 0.95286, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Perihelion_Distance[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Perihelion_Distance[train_balanced$Hazardous == 1]
## W = 0.94887, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Perihelion_Arg[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Perihelion_Arg[train_balanced$Hazardous == 0]
## W = 0.94929, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Perihelion_Arg[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Perihelion_Arg[train_balanced$Hazardous == 1]
## W = 0.95105, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Mean_Anomaly[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Mean_Anomaly[train_balanced$Hazardous == 0]
## W = 0.94659, p-value < 2.2e-16
```

```r
shapiro.test(train_balanced$Mean_Anomaly[train_balanced$Hazardous==1])
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Mean_Anomaly[train_balanced$Hazardous == 1]
## W = 0.9532, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Mean_Motion[train_balanced$Hazardous==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Mean_Motion[train_balanced$Hazardous == 0]
## W = 0.96431, p-value < 2.2e-16
```

```
shapiro.test(train_balanced$Mean_Motion[train_balanced$Hazardous==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_balanced$Mean_Motion[train_balanced$Hazardous == 1]
## W = 0.95362, p-value < 2.2e-16
```

In order to try to obtain artificially the normality for some variables, we have also tried to apply some transformations to the same features. In particular, we tried the *Linear Scaling* method and the *Logarithmic* function, but the results remain the same: the hypothesis of normality is rejected by the tests.

Although this assumption is not satisfied, we try in any case to apply the models to our data, aware of the potential gaps that these could show.

Linear Discriminant Analysis

With Linear Discriminant Analysis (LDA), we aim to find a linear combination of features that characterizes the two classes *Hazardous* and *Not Hazardous*, where the resulting combination will be used as a linear classifier.

In the application of the LDA method we have to take in account that it works when the measurements are made on independent variables. For that reason, the model that will be our starting point will not consider the variables with which we have the problem of collinearity: we re-use the information obtained in the previous models. (The same holds for the Quadratic Discriminant Analysis that we will see in the next section).

We also know, that this type of analysis is quite sensitive to the presence of *outliers*, so in this case we proceed try to find them, looking at the variables summary and boxplots, and we will consider their removal. Looking at the distribution of the variables, we decide to remove the most extreme outlier values and to do that we use the following code:

```
# We consider the previous output given by the summary of the dataset.


# We report here some examples of outliers removal. Then we will
# define the new training dataset without these examples.

# Absolute Magnitude

g1<- ggplot(data = train, aes(y = Absolute_Magnitude,fill = 2)) +
       geom_boxplot(outlier.colour = "red", outlier.shape = 16,
                    outlier.size = 2)+
       theme(legend.position="none") +
       ylab("Absolute Magnitude")
```

```r
# We look for the presence of outliers:

chisq.out.test(train$Absolute_Magnitude)
```

```
##
##  chi-squared test for outlier
##
## data:  train$Absolute_Magnitude
## X-squared = 14.64, p-value = 0.0001301
## alternative hypothesis: lowest value 11.16 is an outlier
```

```r
# Removal of the found outlier:

which(train$Absolute_Magnitude == 11.16) # 256
```

```
## [1] 256
```

```r
# We do the same:

g2<- ggplot(data = train, aes(y = Est_Dia_in_KM_max,fill = 2)) +
       geom_boxplot(outlier.colour = "red", outlier.shape = 16,
                    outlier.size = 2)+
       theme(legend.position="none") +
       ylab("Estimated Diameter")

chisq.out.test(train$Est_Dia_in_KM_max)
```

```
##
##  chi-squared test for outlier
##
## data:  train$Est_Dia_in_KM_max
## X-squared = 1557.2, p-value < 2.2e-16
## alternative hypothesis: highest value 34.836938254 is an outlier
```

```r
which(train$Est_Dia_in_KM_max == 34.836938254) # 256
```

```
## [1] 256
```

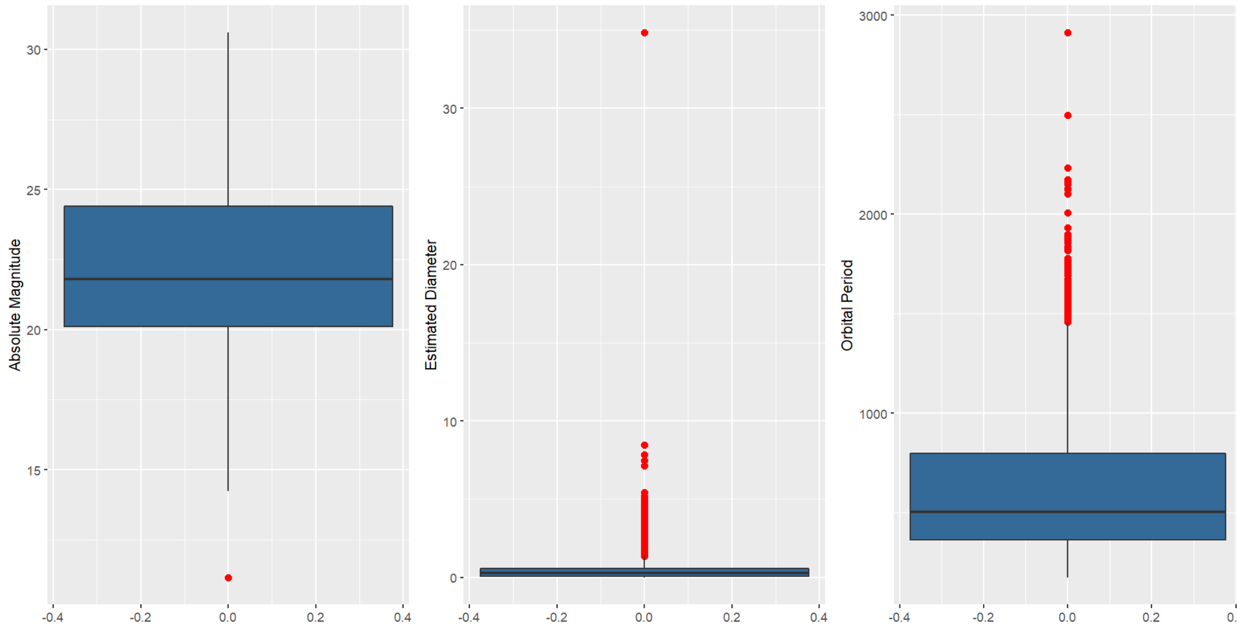
```r
g3<- ggplot(data = train, aes(y = Orbital_Period,fill = 2)) +
       geom_boxplot(outlier.colour = "red", outlier.shape = 16,
                    outlier.size = 2)+
       theme(legend.position="none") +
       ylab("Orbital Period")

chisq.out.test(train$Orbital_Period)
```

```
##
##  chi-squared test for outlier
##
## data:  train$Orbital_Period
## X-squared = 38.353, p-value = 5.903e-10
## alternative hypothesis: highest value 2912.0220196159 is an outlier
```

```r
which(train$Orbital_Period == 2912.0220196159) # 1556
```

```
## [1] 1556
```

```r
grid.arrange(g1, g2, g3, nrow = 1)
```



Another note needs to be done regarding a possible "features selection" with the followings models. For LDA we can easily observe the values of the coefficients estimated for each variable, while for the QDA it's more difficult. For that reason we will consider all the set of variables.

Simple LDA

We define now a *Simple LDA model*, as we said before, starting from the knowledge about the best subset of variables to use deriving from the analysis conduced with the use of *VIF*.

```r
library(MASS)

# Model definition:

lda_compl<- lda(Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis -
    Jupiter_Tisserand_Invariant - Eccentricity - Mean_Motion -
    Est_Dia_in_KM_max, family = "binomial", data = train)

# Observing the model summary:

lda_compl
```

```
## Call:
## lda(Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis - Jupiter_Tisserand_Invariant -
##     Eccentricity - Mean_Motion - Est_Dia_in_KM_max, data = train,
##     family = "binomial")
##
## Prior probabilities of groups:
##         0         1
## 0.8420154 0.1579846
##
## Group means:
##   Absolute_Magnitude Relative_Velocity_km_per_hr Miss_Dist_Astronomical
```

```
## 0           22.63252                    48559.65              0.2566026
## 1           20.12481                    61828.79              0.2651441
##   Orbit_Uncertainity Minimum_Orbit_Intersection Inclination Asc_Node_Longitude
## 0           4.920554                 0.09481278    13.53324           169.9012
## 1           2.216216                 0.02291273    13.15031           176.2912
##   Orbital_Period Perihelion_Distance Perihelion_Arg Perihelion_Time
## 0       632.8233            1821.232       185.2189         2457714
## 1       637.5504            1222.000       185.2448         2457815
##   Mean_Anomaly
## 0     178.9938
## 1     194.5807
##
## Coefficients of linear discriminants:
##                                       LD1
## Absolute_Magnitude          -3.381515e-01
## Relative_Velocity_km_per_hr  1.543152e-06
## Miss_Dist_Astronomical       4.150228e-01
## Orbit_Uncertainity          -1.106080e-01
## Minimum_Orbit_Intersection  -1.348479e+01
## Inclination                 -3.463805e-03
## Asc_Node_Longitude           1.046913e-04
## Orbital_Period               4.042363e-04
## Perihelion_Distance          1.202690e-05
## Perihelion_Arg              -1.265689e-04
## Perihelion_Time             -4.626450e-05
## Mean_Anomaly                 3.524736e-04
```

```r
# Computing predictions:

pred_lda_compl<- predict(lda_compl, test, type = "response") # threshold: 0.5
post_lda_compl<- pred_lda_compl$posterior

# Converting the predictions in {0,1} according to the chosen threshold:

pred_lda_compl_04<- as.factor(ifelse(post_lda_compl[,2] > threshold4, 1, 0))
pred_lda_compl_05<- pred_lda_compl$class
pred_lda_compl_06<- as.factor(ifelse(post_lda_compl[,2] > threshold6, 1, 0))
```

From the call to the summary method what we see is that the variables that show greatest coefficients are:

- Absolute Magnitude
- Missing Distance
- Orbit Uncertainity
- Minimum Orbit Intersection

We see now how the combination of these variables can perform the classification.

```r
# Confusion matrix with threshold = 0.4

table(test$Hazardous, pred_lda_compl_04)
mean(pred_lda_compl_04!=test$Hazardous)

# Confusion matrix with threshold = 0.5

table(test$Hazardous, pred_lda_compl_05)
mean(pred_lda_compl_05!=test$Hazardous)
```

```
# Confusion matrix with threshold = 0.6

table(test$Hazardous, pred_lda_compl_06)
mean(pred_lda_compl_06!=test$Hazardous)
```

With the same thresholds used until now, the results obtained are worst in terms of false negative. We can try less values and to observe if we are able to make better the model's performance:

```
pred_lda_compl_02<- as.factor(ifelse(post_lda_compl[,2] > 0.2, 1, 0))
pred_lda_compl_03<- as.factor(ifelse(post_lda_compl[,2] > 0.3, 1, 0))

# Confusion matrix with threshold = 0.2

table(test$Hazardous, pred_lda_compl_02)
```

```
##    pred_lda_compl_02
##       0   1
##   0 857 115
##   1   9 191
```

```
mean(pred_lda_compl_02!=test$Hazardous)
```

```
## [1] 0.105802
```

```
# Confusion matrix with threshold = 0.3

table(test$Hazardous, pred_lda_compl_03)
```

```
##    pred_lda_compl_03
##       0   1
##   0 889  83
##   1  31 169
```

```
mean(pred_lda_compl_03!=test$Hazardous)
```

```
## [1] 0.09726962
```

Confusion Matrices - Simple LDA

Treshold: 0.3			
	Predictions		
Real Values	0	1	Total
0	889	83	972
1	31	169	200
Total	920	252	1172

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	916	56	972
1	48	152	200
Total	964	208	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	931	41	972
1	72	128	200
Total	1003	169	1172

We can observe here that we have two good results: with thresholds 0.3 and 0.4. If we analyze better the matrices, we can see that for the first the general error is greater but we have a less rate of *False Negative*. In the other case, instead we have a slightly less general error but the rate of *False Negative* is higher. Considering our aim for the classification of dangerous bodies for the Earth, in this case we are most interested in a model like the one with the threshold 0.3.

```
# We use now the information given by:
# - x: linear combination of the variables that better describe the examples
# - class: assigned class

ldahist(pred_lda_compl$x[,1], g = pred_lda_compl$class, col = 2)
```

The results of the linear discriminant analysis are well explained in the above graph. This represents the values of the discriminant function previously calculated for the observations of the two different groups

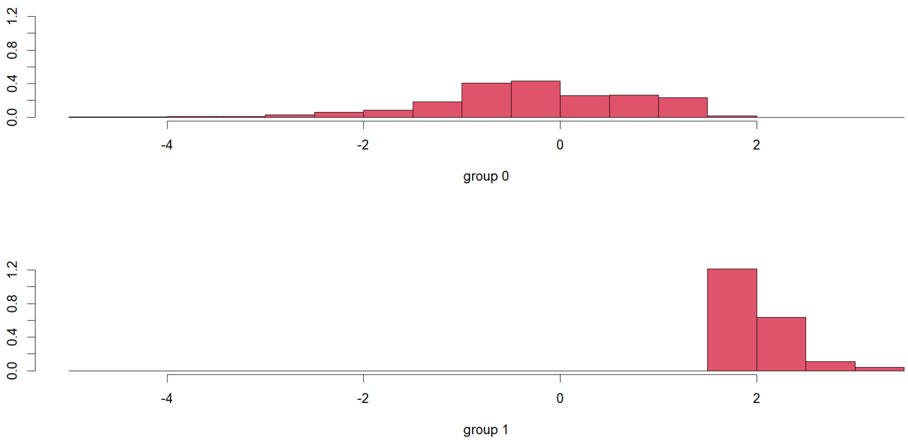


Figure 4: Histograms of how the combinations of variables classify the examples

(Hazardous and Not Hazardous). We can see that the function discriminates the two groups quite well, having only an overlap when it assumes greater values, which however belong more to the class of dangerous celestial bodies.

LDA with balanced data

We apply now the same process on the balanced dataset.

```
# We consider the previous output given by the summary of the dataset.

# We report here some examples of outliers removal. Then we will
# define the new training dataset without these examples.

# Absolute Magnitude

g4<-ggplot(data = train_balanced, aes(y =     Relative_Velocity_km_per_hr,fill = 2)) +
       geom_boxplot(outlier.colour = "red", outlier.shape = 16,
                    outlier.size = 2)+
       theme(legend.position="none") +
       ylab("Relative Velocity")

# We look for the presence of outliers:

chisq.out.test(train_balanced$Relative_Velocity_km_per_hr)
```

```
##
##  chi-squared test for outlier
##
## data:  train_balanced$Relative_Velocity_km_per_hr
## X-squared = 14.35, p-value = 0.0001518
## alternative hypothesis: highest value 160681.487851189 is an outlier
```

```
# Removal of the found outlier:

which(train_balanced$Relative_Velocity_km_per_hr >= 160681.487851189) # 2313 2580
```

```
## [1] 2313 2580
```

```r
# We do the same:

g5<- ggplot(data = train_balanced, aes(y = Est_Dia_in_KM_max,fill = 2)) +
       geom_boxplot(outlier.colour = "red", outlier.shape = 16,
                    outlier.size = 2)+
       theme(legend.position="none") +
       ylab("Estimated Diameter")

chisq.out.test(train_balanced$Est_Dia_in_KM_max)
```

```
##
##  chi-squared test for outlier
##
## data:  train_balanced$Est_Dia_in_KM_max
## X-squared = 1066.4, p-value < 2.2e-16
## alternative hypothesis: highest value 34.836938254 is an outlier
```

```r
which(train_balanced$Est_Dia_in_KM_max == 34.836938254) # 1301 1703
```

```
## [1] 1301 1703
```

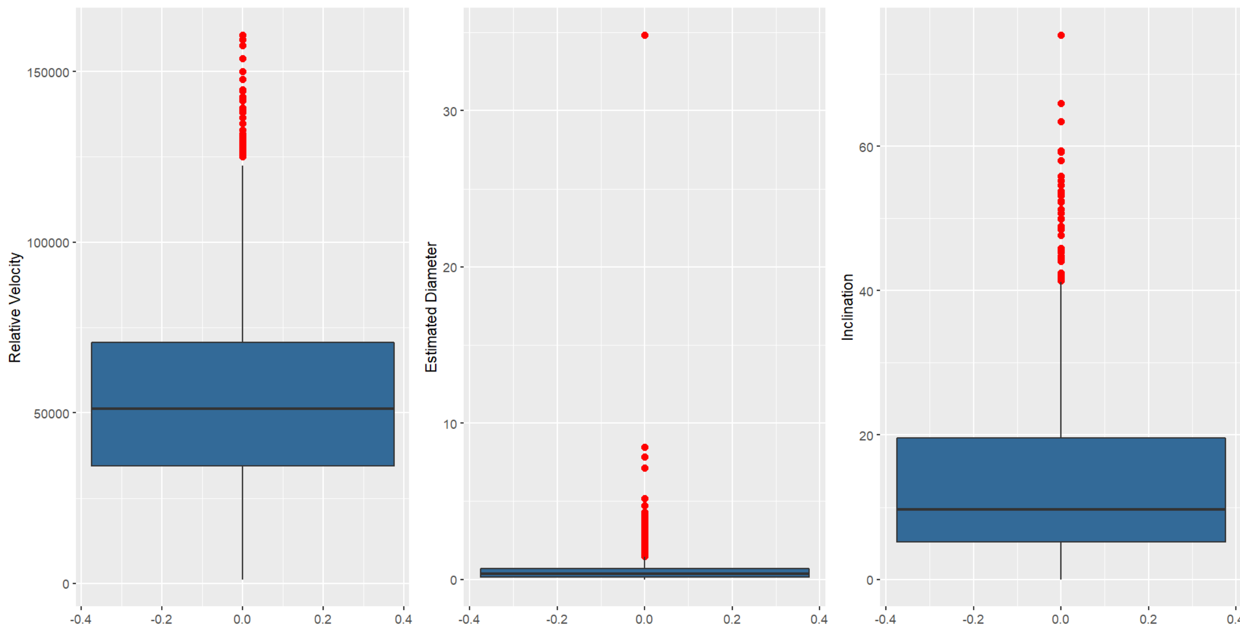
```r
g6<- ggplot(data = train_balanced, aes(y = Inclination,fill = 2)) +
       geom_boxplot(outlier.colour = "red", outlier.shape = 16,
                    outlier.size = 2)+
       theme(legend.position="none") +
       ylab("Inclination")

chisq.out.test(train_balanced$Inclination)
```

```
##
##  chi-squared test for outlier
##
## data:  train_balanced$Inclination
## X-squared = 32.87, p-value = 9.855e-09
## alternative hypothesis: highest value 75.4066668415747 is an outlier
```

```r
which(train_balanced$Inclination >= 75.406666841) # 3044
```

```
## [1] 3044
```

```r
grid.arrange(g4, g5, g6, nrow = 1)
```




```r
# Model definition starting from the previous glm_bal model:

lda_bal<- lda(data = train_balanced,
              Hazardous ~.-Aphelion_Dist-Semi_Major_Axis-
              Jupiter_Tisserand_Invariant-Eccentricity-
              Est_Dia_in_KM_max-Mean_Motion,
              family = "binomial")

# Observing the model summary:

lda_bal
```

```
## Call:
## lda(Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis - Jupiter_Tisserand_Invariant -
##     Eccentricity - Est_Dia_in_KM_max - Mean_Motion, data = train_balanced,
##     family = "binomial")
##
## Prior probabilities of groups:
##         0         1
## 0.5145299 0.4854701
##
## Group means:
##   Absolute_Magnitude Relative_Velocity_km_per_hr Miss_Dist_Astronomical
## 0           22.57911                    48769.74              0.2577234
## 1           20.14377                    61401.26              0.2669771
##   Orbit_Uncertainity Minimum_Orbit_Intersection Inclination Asc_Node_Longitude
## 0           4.901440                 0.09352232    13.38186           169.2607
## 1           2.224765                 0.02315230    13.05561           175.0857
##   Orbital_Period Perihelion_Distance Perihelion_Arg Perihelion_Time
## 0       649.4341            1846.770       187.7336         2457729
## 1       630.3585            1209.519       181.7973         2457841
##   Mean_Anomaly
## 0     177.8747
## 1     198.5249
```

```
##
## Coefficients of linear discriminants:
##                                       LD1
## Absolute_Magnitude          -3.645904e-01
## Relative_Velocity_km_per_hr -2.205457e-06
## Miss_Dist_Astronomical       2.588598e-01
## Orbit_Uncertainity          -1.418070e-01
## Minimum_Orbit_Intersection  -1.658431e+01
## Inclination                  4.660338e-03
## Asc_Node_Longitude          -1.553317e-04
## Orbital_Period               5.950468e-04
## Perihelion_Distance         -1.145857e-04
## Perihelion_Arg              -6.019002e-04
## Perihelion_Time             -1.258085e-05
## Mean_Anomaly                 1.633629e-04
```

```r
# Computing the predictions with the model on the test set:

pred_lda_bal<- predict(lda_bal, test, type = "response")
post_lda_bal<- pred_lda_bal$posterior


# Converting the predictions in {0,1} according to the chosen threshold:

pred_lda_bal_03<- as.factor(ifelse(post_lda_bal[,2] > 0.3, 1, 0))
pred_lda_bal_04<- as.factor(ifelse(post_lda_bal[,2] > threshold4, 1, 0))
pred_lda_bal_05<- pred_lda_bal$class
pred_lda_bal_06<- as.factor(ifelse(post_lda_bal[,2] > threshold6, 1, 0))
```

With the use of the balanced dataset, the variables that most influence the discrimination are:

- Absolute Magnitude
- Missing Distance
- Orbit Uncertainity
- Minimum Orbit Intersection

and they are the same as before.

Let's see if the confusion matrices can give us more details about the classification performed by this model.

```r
# Confusion matrix with threshold: 0.3

table(test$Hazardous, pred_lda_bal_03)
mean(pred_lda_bal_03!=test$Hazardous)

# Confusion matrix with threshold: 0.4

table(test$Hazardous, pred_lda_bal_04)
mean(pred_lda_bal_04!=test$Hazardous)

# Confusion matrix with threshold: 0.5

table(test$Hazardous, pred_lda_bal_05)
mean(pred_lda_bal_05!=test$Hazardous)

# Confusion matrix with threshold: 0.6
```

```
table(test$Hazardous, pred_lda_bal_06)
mean(pred_lda_bal_06!=test$Hazardous)

ldahist(pred_lda_bal$x[,1], g = pred_lda_bal$class, col = 2)
```

Confusion Matrices - LDA with Balanced dataset

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	797	175	972
1	6	194	200
Total	803	369	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	835	137	972
1	6	194	200
Total	841	331	1172

Treshold: 0.6			
	Predictions		
Real Values	0	1	Total
0	867	105	972
1	11	189	200
Total	878	294	1172

Althought we computed the predictions also for the threshold 0.3, here we have better results with the other values. With the considered thresholds we have in any case the *False Negative* rate less than the *False Positive* one. The best result here is obtained with the threshold equal to 0.6: the total error is the less one, but we have an increased error on the classification of the hazardous element as *Not Hazardous*. If we compare the differences between the two errors, we may agree that the difference between the two false negative rate is less than the difference in terms of general error: in this case we prefer the last classification.

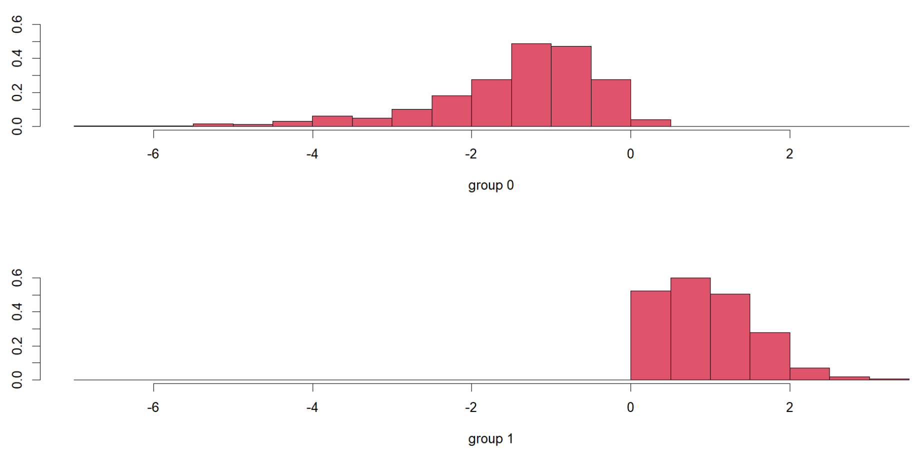


Figure 5: Histograms of how the combinations of variables classify the examples

Even in the case of balanced data, through the following graph we can see that the function discriminates quite well the two classes, associating the class of dangerous celestial bodies to values greater than 0.

Quadratic Discriminant Analysis

We implement now also a model using the *QDA* method with which we should be able to observe better results in general than *LDA*: LDA, in fact, tends to works well where there are few training observations and for that we need to estimate fewer parameters. In this case it's also important to reduce the variance. In the other hand, if the set of observations is larger, the amount of variance is not relevant: we can use QDA in order to have the possibility to estimate more parameters and to have a more flexible model.

Simple QDA

```
# Model definition starting from the previous glm_compl:

qda_compl<- qda(Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis -
```

```r
    Jupiter_Tisserand_Invariant - Eccentricity - Mean_Motion -
    Est_Dia_in_KM_max- Orbit_Uncertainity,
    family = "binomial", data = train)


# Computing predictions:

pred_qda_compl<- predict(qda_compl, test, type = "response") # threshold: 0.5
post_qda_compl<- pred_qda_compl$posterior


# Converting the predictions in {0,1} according to the chosen threshold:

pred_qda_compl_03<- as.factor(ifelse(post_qda_compl[,2] > 0.3, 1, 0))
pred_qda_compl_04<- as.factor(ifelse(post_qda_compl[,2] > threshold4, 1, 0))
pred_qda_compl_05<- pred_qda_compl$class
pred_qda_compl_06<- as.factor(ifelse(post_qda_compl[,2] > threshold6, 1, 0))
```

```r
# Confusion matrix with threshold: 0.3

table(test$Hazardous, pred_qda_compl_03)
mean(pred_qda_compl_03!=test$Hazardous)

# Confusion matrix with threshold: 0.4

table(test$Hazardous, pred_qda_compl_04)
mean(pred_qda_compl_04!=test$Hazardous)

# Confusion matrix with threshold: 0.5

table(test$Hazardous, pred_qda_compl_05)
mean(pred_qda_compl_05!=test$Hazardous)

# Confusion matrix with threshold: 0.6

table(test$Hazardous, pred_qda_compl_06)
mean(pred_qda_compl_06!=test$Hazardous)
```

From this confusion matrices, we note that considering high threshold as for example 0.6, the rate of *False Positive* becomes less than the *False Negatives*. In order to compare good results for the *Simple QDA* model we consider less thresholds, starting also here from 0.3 up to 0.5 .

Confusion Matrices - Simple QDA

Treshold: 0.3			
	Predictions		
Real Values	0	1	Total
0	933	39	972
1	6	194	200
Total	939	233	1172

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	944	28	972
1	8	192	200
Total	952	220	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	954	18	972
1	17	183	200
Total	971	201	1172

The values that we can observe with QDA are considerable: we have a very low value of general error and at the same time the rate of *False Negative* is the less error present from the model. In particular, if we have to choose the best option for the threshold, here we will choose 0.4

QDA with balanced data

```r
# Model definition starting from the previous glm_bal model:

qda_bal<- qda(data = train_balanced,
              Hazardous ~.-Aphelion_Dist-Semi_Major_Axis-
              Jupiter_Tisserand_Invariant-Eccentricity-
              Est_Dia_in_KM_max-Mean_Motion- Orbit_Uncertainity,
              family = "binomial")

# Observing the model summary:

qda_bal
```

```
## Call:
## qda(Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis - Jupiter_Tisserand_Invariant -
##     Eccentricity - Est_Dia_in_KM_max - Mean_Motion - Orbit_Uncertainity,
##     data = train_balanced, family = "binomial")
##
## Prior probabilities of groups:
##         0         1
## 0.5145299 0.4854701
##
## Group means:
##   Absolute_Magnitude Relative_Velocity_km_per_hr Miss_Dist_Astronomical
## 0           22.57911                    48769.74              0.2577234
## 1           20.14377                    61401.26              0.2669771
##   Minimum_Orbit_Intersection Inclination Asc_Node_Longitude Orbital_Period
## 0                 0.09352232    13.38186           169.2607       649.4341
## 1                 0.02315230    13.05561           175.0857       630.3585
##   Perihelion_Distance Perihelion_Arg Perihelion_Time Mean_Anomaly
## 0            1846.770       187.7336         2457729     177.8747
## 1            1209.519       181.7973         2457841     198.5249
```

```r
# Computing the predictions with the model on the test set:

pred_qda_bal<- predict(qda_bal, test, type = "response")
post_qda_bal<- pred_qda_bal$posterior


# Converting the predictions in {0,1} according to the chosen threshold:

pred_qda_bal_03<- as.factor(ifelse(post_qda_bal[,2] > 0.3, 1, 0))
pred_qda_bal_04<- as.factor(ifelse(post_qda_bal[,2] > threshold4, 1, 0))
pred_qda_bal_05<- pred_qda_bal$class
pred_qda_bal_06<- as.factor(ifelse(post_qda_bal[,2] > threshold6, 1, 0))
```

```r
# Confusion matrix with threshold: 0.3

table(test$Hazardous, pred_qda_bal_03)
mean(pred_qda_bal_03!=test$Hazardous)

# Confusion matrix with threshold: 0.4

table(test$Hazardous, pred_qda_bal_04)
```

```
mean(pred_qda_bal_04!=test$Hazardous)

# Confusion matrix with threshold: 0.5

table(test$Hazardous, pred_qda_bal_05)
mean(pred_qda_bal_05!=test$Hazardous)

# Confusion matrix with threshold: 0.6

table(test$Hazardous, pred_qda_bal_06)
mean(pred_qda_bal_06!=test$Hazardous)
```

Confusion Matrices - QDA with Balanced dataset

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	875	97	972
1	3	197	200
Total	878	294	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	892	80	972
1	4	196	200
Total	896	276	1172

Treshold: 0.6			
	Predictions		
Real Values	0	1	Total
0	915	57	972
1	5	195	200
Total	920	252	1172

Also in this case we have good results, but we can observe that the total error is generally increased. In the other hand, the *False Negative* rate is almost inexistent, and this brings the QDA model to be considered a good solution for our classification task.

Regularized Regression

In this section we fit generalized models where we add some *penalizations*, introducing *Lasso* and *Ridge* regularizations. They are two different statistical methods that allow us to compute an automatic selection of variables, operating shrinkage on the coefficients of the predictors in such a way that they assume values very close to zero or even zero. When we have estimators that tend to have very large variants and small bias, we can also have presence of multicollinearity. However, estimators that have very large variants will produce poor estimates. This phenomenon is referred to as *Overfitting*.

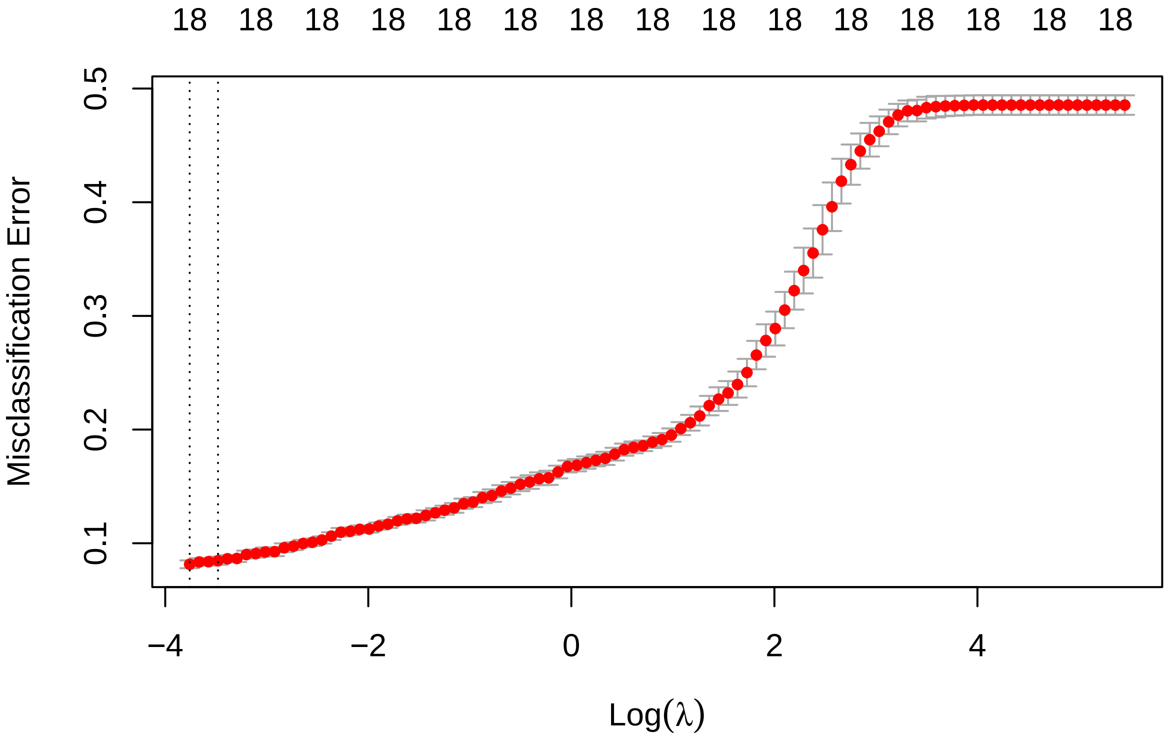
According to the results obtained until now, we decide to continue to test our model only on the balanced dataset.

Regarding the features selection step, we need to do some considerations. For the *Lasso* model we have not to implement it because, thanks to its formulation, it does variable selection in the process of shrinking to zero the values of coefficients of variables. For the *Ridge* we have basically the same behaviour with the only difference that here the coefficients are not directly set to zero, as in the lasso, but shrinked more and more until reaching zero.

Ridge Regression

```
# We use here the balanced training dataset because we have seen that, generally, we
# obtain better results.

train_bal_mat<- as.matrix(train_balanced[,-19])
test_mat<- as.matrix(test[,-19])

# We look for the best value for lambda in order to define the best model:

# We use cross validation glmnet

ridge_cv <- cv.glmnet(train_bal_mat, train_balanced$Hazardous,
                      alpha = 0, family = "binomial", type.measure = "class")
```

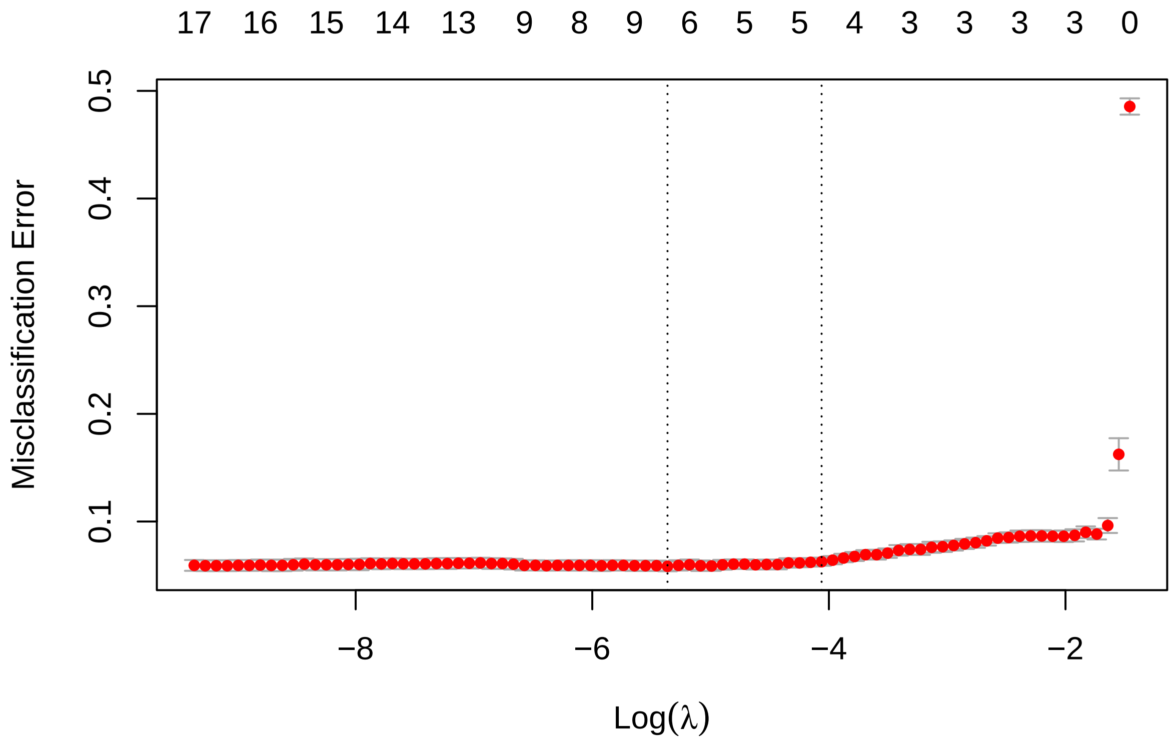
```
plot(ridge_cv)
```




```
# We identify th best lambda value

lambda_opt_ridge <- ridge_cv$lambda.min
lambda_opt_ridge

# We compute predictions with the ridge model using the best value for
# lambda that we have obtained

pred_ridge<- predict(ridge_cv, test_mat, type = "class", s = lambda_opt_ridge)

table(test$Hazardous, pred_ridge)
```

Lasso Regression

```
# We look for the best value for lambda in order to define the best model:

# We use cross validation glmnet

lasso_cv <- cv.glmnet(train_bal_mat, train_balanced$Hazardous,
                      alpha = 1, family = "binomial", type.measure = "class")

plot(lasso_cv)
```




```r
# We identify th best lambda value

lambda_opt_lasso <- lasso_cv$lambda.min
lambda_opt_lasso

# We compute predictions with the lasso model using the best value for
# lambda that we have obtained

pred_lasso<- predict(lasso_cv, test_mat, type = "class", s = lambda_opt_lasso)

table(test$Hazardous, pred_lasso)
```

Ridge Regression			
	Predictions		
Real Values	-1	1	Total
-1	863	109	972
1	9	191	200
Total	872	300	1172

Lasso Regression			
	Predictions		
Real Values	-1	1	Total
-1	910	62	972
1	8	192	200
Total	918	254	1172

We can now compare the performances of these two regularization methods. As the matrices explain, using the *Lasso* alternative, we are able to reach both a total error and a *False Negative* rate less than using the *Ridge* model.

K-Nearest Neighbors

The *K-Nearest Neighbor* (KNN) algorithm is a completely non-parametric approach that represents one of the most widely used algorithms. In particular, here, no assumptions are made about the shape of the decision boundary. To make a prediction for an observation x using *KNN* we need to:

- identify the k training observations that are closest to x;
- assigning x to the class to which the plurality of these observations belong.

```r
# Function for linear scaling our features

min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

# We normalize the columns

nasa_n <- as.data.frame(lapply(nasa[,-19], min_max_norm))

# We re-add the Hazardous features to re-complete the dataset

nasa_n$Hazardous <- nasa$Hazardous

# We re-do the split obtaining the same differentiation as before

set.seed(0607)

split_n <- initial_split(nasa_n, prop = 0.75)
train_n <- training(split_n)
test_n <- testing(split_n)

# We balance again our dataset

train_n_balanced<- ovun.sample(Hazardous~., data = train_n,
                             method = "under", p = 1/5.35,
                              seed = 1)$data

# We implement a first KNN model removing the variables that show collinearity

library(class)

# We look now for the best value of the parameter k

kmax <- 100

test_error <- numeric(kmax)

# For each possible value of k we consider the obtained accuracy of the model

for (k in 1:kmax) {
  knn_pred <- as.factor(knn(train_n_balanced[,-c(2,7,8,9,15,18,19)],
                            test_n[,-c(2,7,8,9,15,18,19)],
                            cl = train_n_balanced$Hazardous, k = k))

  cm <- confusionMatrix(data = knn_pred, reference = as.factor(test_n$Hazardous))

  test_error[k] <- 1 - cm$overall[1]
}

# We took the minimum value of the error

k_min <- which.min(test_error)
```

```r
# We compute now the prediction with the value of k that gives us the minimum error

knn<- knn(train_n_balanced[,-c(2,7,8,9,15,18,19)], test_n[,-c(2,7,8,9,15,18,19)],
          cl = train_n_balanced$Hazardous, k = k_min)

knn_pred_min <- as.factor(knn)

# Confusion matrix for KNN on the test set

tab<- table(test_n$Hazardous, knn)
tab
```

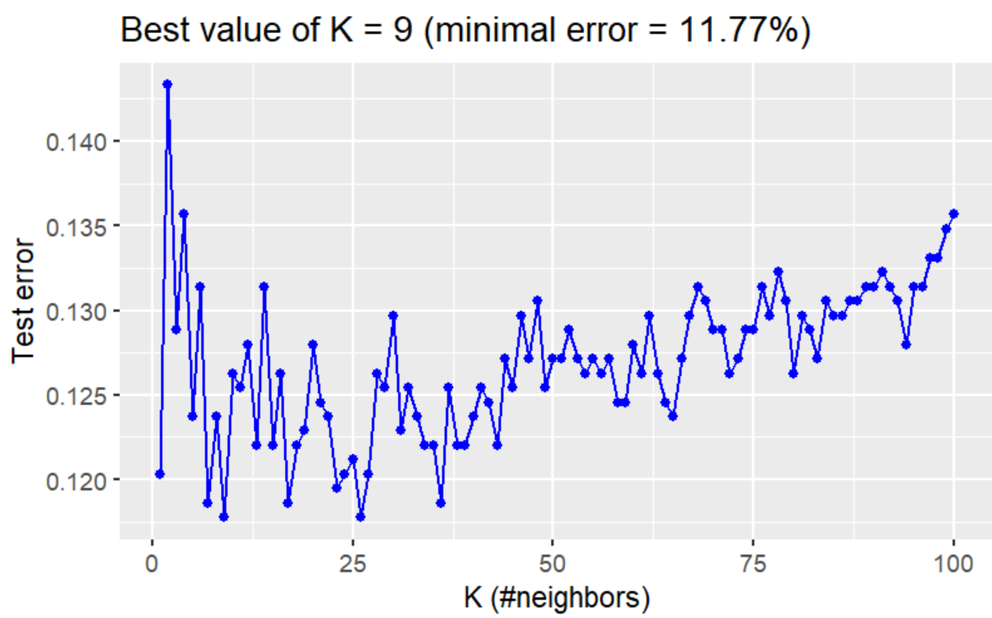
```
##    knn
##       0   1
##   0 920  52
##   1  86 114
```

```r
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)
```

```
## [1] 88.22526
```

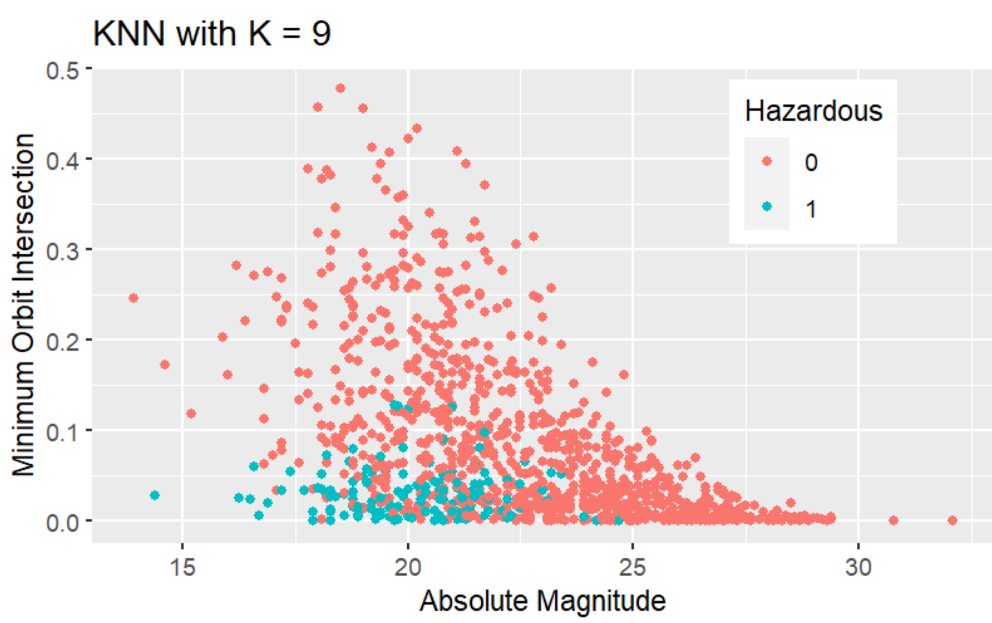
```r
cm <- confusionMatrix(data = knn_pred_min, reference = as.factor(test_n$Hazardous))

ggplot(data.frame(test_error), aes(x = 1:kmax, y = test_error)) +
  geom_line(colour="blue") +
  geom_point(colour="blue") +
  xlab("K (#neighbors)") + ylab("Test error") +
  ggtitle(paste0("Best value of K = ", k_min,
                 " (minimal error = ",
                 format((test_error[k_min])*100, digits = 4), "%)"))
```

```r
ggplot(test, aes(x = Absolute_Magnitude,
                     y = Minimum_Orbit_Intersection,
                     color = as.factor(knn))) +
  geom_point()+
  labs(x = "Absolute Magnitude",
       y = "Minimum Orbit Intersection",
       color = "Hazardous",
       title = "KNN with K = 9")  +
  theme(legend.position = c(0.8, 0.8))
```



To achieve the best results we can get from knn, we will look for the optimal value of K by calculating the error rate value on the test data on multiple values of K and choosing the value that minimizes that error.

K = 9 seems to provide the lowest test error rate (around 11%) so we choose this as the value for K to evaluate our classification. We can represent the resulting classification in the image below.



How we can see, the classification done with this method is far from the ones obtained with the other models. In particular, comparing the plot with the true values of *Hazardous* and *Not Hazardous*, we observe that we have points classified as dangerous with not a "clear decision boundary" defined by the same model, while in the original classification we have like a "unique block" of dangerous bodies. This brings the model to have such a bad result.

Model Considerations

We can now summarize the obtained results with all the types of models we have implemented, choosing particular metrics in order to compare them and their performances on the same level. Referring to the nature of the presented problem of classification of hazardous NEOs for the Earth, we decide to consider the *Overall Error* rate and at the same time the *False Negative* rate. Doing in this way we are to understand the general correctness of the predictions of the models and their safety: as we said in the initial presentation, we prefer, in fact, to have an higher proportion of *Not Hazardous* bodies classified as *Hazardous* instead of higher quantities of *Hazardous* classified as *Not Hazardous*. As we can see in particular, the trial with a non-parametric model doesn't bring such a good result, so in the final comparison we will see only the previous models.

Comparison of Models

Model	Overall Error Rate (FalsePositive + FalseNegative)/(Total)	False Negative Rate FalseNegative/(FalseNegative+TruePositive)
Simple GLM - 0.4	0,055	0,135
Simple GLM - 0.5	0,052	0,175
Simple GLM - 0.6	0,058	0,24
GLM with Stepwise - 0.4	0,054	0,125
GLM with Stepwise - 0.5	0,053	0,175
GLM with Stepwise - 0.6	0,056	0,23
GLM with Balanced dataset - 0.4	0,095	0,035
GLM with Balanced dataset - 0.5	0,079	0,055
GLM with Balanced dataset - 0.6	0,067	0,075
GLM with Balanced dataset and Stepwise - 0.4	0,095	0,035
GLM with Balanced dataset and Stepwise - 0.5	0,083	0,065
GLM with Balanced dataset and Stepwise - 0.6	0,067	0,08
Simple LDA - 0.3	0,096	0,15
Simple LDA - 0.4	0,089	0,24
Simple LDA - 0.5	0,096	0,365
LDA with Balanced dataset - 0.3	0,181	0,025
LDA with Balanced dataset - 0.4	0,154	0,03
LDA with Balanced dataset - 0.5	0,122	0,055
Simple QDA - 0.3	0,037	0,03
Simple QDA - 0.4	0,029	0,04
Simple QDA - 0.5	0,029	0,085
QDA with Balanced dataset - 0.4	0,085	0,015
QDA with Balanced dataset - 0.5	0,072	0,02
QDA with Balanced dataset - 0.6	0,053	0,025
Ridge Regression - 0.55	0,101	0,045
Lasso Regression - 0.15	0,059	0,04
KNN	0,118	0,43

From the summarizing table we can do some considerations:

- using the balancing of the dataset we are able to improve the results of the model in terms of *False Negative*, while we can observe an increasing of the *Overall Error* rate;
- the use of the *Stepwise* approach doesn't bring improvements of goodness of the classification;
- for the GLM method, the best rates are given by the Logistic Regression applied on the balanced dataset and using the threshold equal to 0.4: considering a slightly less threshold against the classic 0.5, we optimize for recall;

- for LDA we choose the model trained on the balanced data for which we classify the observation using 0.5 as threshold: in this case, in fact, we consider the option with less value of the general error, beacuse the difference in terms of false negative rate is not so relevant;
- for QDA we chose the model that is trained on the original dataset and uses as threshold 0.3. Also the model trained on the balanced dataset and 0.6 as value could be a good option: also in this case, in order to choose the best one between them, we consider the difference in terms of both overall error rate and false negative rate. The greatest distance between the proposed model is about the general error, so we decide to maintain QDA trained on the unbalanced dataset;
- between the two regularization methods, we can observe that Lasso is the best performing one, with half of overall error rate compared with the error computed by Ridge regression;
- at the end, we consider the results obtained with KNN method. Our aim is to minimize both the overall error rate and the false negative rate: in this case we have an error of classification of the hazardous NEOs that's around 0.40. This issue is produced by the use of a low value for the hyperparamer k with which we have the highest accuracy score.

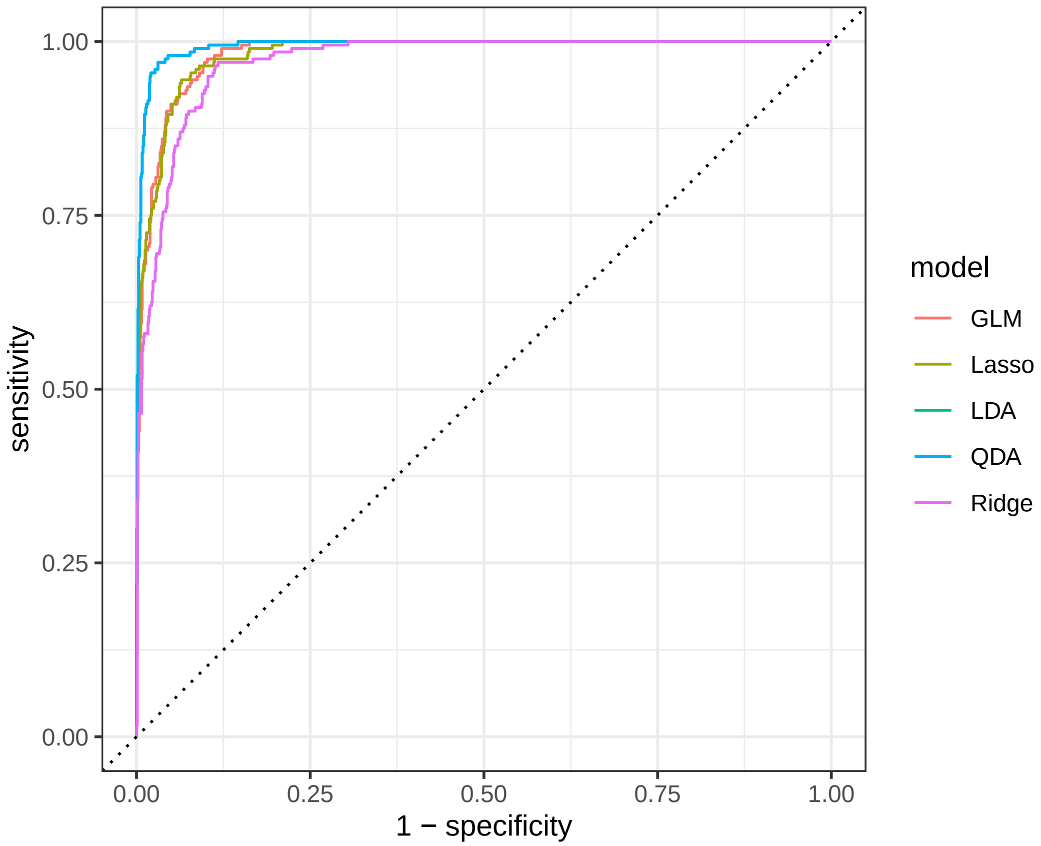
ROC Curve

We now introduce the ROC curve tool, a curve capable of showing the performance of a model, which is obtained plotting the *False Positive Rate*, on the x-axis, and *True Positive Rate*, on y-axis, on the Cartesian reference system. In order to understand which is the best model, we consider the area under the curve: greater is the area, better is the model's performance.

```r
# Best GLM model

glm_best<- glm(data = train_balanced,
              Hazardous ~ Absolute_Magnitude+Minimum_Orbit_Intersection+
                Orbit_Uncertainity+Orbital_Period,
              family = "binomial")

pred_glm_best<- predict(glm_best, test, type = "response")

# Best LDA model

lda_best<- lda(data = train_balanced,
              Hazardous ~Absolute_Magnitude+Miss_Dist_Astronomical+
                Orbit_Uncertainity+Minimum_Orbit_Intersection,
              family = "binomial")

pred_lda_best<- predict(lda_best, test, type = "response")
post_lda_best<- pred_lda_best$posterior

# Best QDA model

qda_best<- qda(Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis -
    Jupiter_Tisserand_Invariant - Eccentricity - Mean_Motion -
    Est_Dia_in_KM_max- Orbit_Uncertainity,
    family = "binomial", data = train)

pred_qda_best<- predict(qda_best, test, type = "response")
post_qda_best<- pred_qda_best$posterior

# Best Ridge model
```

```r
ridge_best<- glmnet(train_bal_mat, train_balanced$Hazardous,
                    alpha = 0, family = "binomial", lambda = lambda_opt_ridge)

pred_ridge_best<- predict(ridge_best, test_mat, type = "response", s = lambda_opt_ridge)

# Best Lasso model

lasso_best<- glmnet(train_bal_mat, train_balanced$Hazardous,
                    alpha = 0, family = "binomial", lambda = lambda_opt_lasso)

pred_lasso_best<- predict(lasso_best, test_mat, type = "response", s = lambda_opt_lasso)
```

```r
# We compare the best models of each type looking at the ROC curve

prediction <- tibble(truth = as.factor(test$Hazardous))
prediction <- prediction %>% mutate(pred = as.numeric(pred_glm_best))%>%
  mutate(model= "GLM")%>%
  add_row(truth = as.factor(test$Hazardous), pred = as.numeric(post_qda_best[,2]),
          model= "LDA")%>%
  add_row(truth = as.factor(test$Hazardous), pred = as.numeric(post_qda_best[,2]),
          model= "QDA")%>%
  add_row(truth = as.factor(test$Hazardous), pred = as.numeric(pred_ridge_best),
          model= "Ridge")%>%
  add_row(truth = as.factor(test$Hazardous), pred = as.numeric(pred_lasso_best),
          model= "Lasso")


roc <- prediction %>% group_by(model) %>%
  roc_curve(truth, pred, event_level = "second") %>%
  autoplot()
roc
```




```
auc(test$Hazardous, pred_glm_best)
```

```
## Area under the curve: 0.9832
```

```
auc(test$Hazardous, post_lda_best[,2])
```

```
## Area under the curve: 0.966
```

```
auc(test$Hazardous, post_qda_best[,2])
```

```
## Area under the curve: 0.9938
```

```
auc(test$Hazardous, pred_ridge_best)
```

```
## Area under the curve: 0.9718
```

```
auc(test$Hazardous, pred_lasso_best)
```

```
## Area under the curve: 0.9824
```

Looking both the ROC-curve plot and the Area Under the Curve (AUC) we can deduce that the best model for our classification task is the QDA. If we have to consider the other models, we are in the position to say that they are all good models in terms of obtained accuracy, but in this case we have to make reference also at the above presented table. Here in fact we can see also the *False Negative* rate computed for each model and we can observe that for QDA, Lasso and Ridge we have the best result over all the other models. For that reason the final choice, driven by the considerations done across the full project are these.

```
e<- ggplot(test, aes(x = Absolute_Magnitude, y =
Minimum_Orbit_Intersection, color = as.factor(ifelse(pred_glm_best > 0.5, 1, 0)))) +
geom_point()+ labs(x = "Absolute Magnitude", y = "Minimum Orbit
Intersection", color = "Hazardous", title = "GLM") + theme(legend.position = c(0.8, 0.8))
```

```
f<- ggplot(test, aes(x = Absolute_Magnitude, y =
Minimum_Orbit_Intersection, color =
as.factor(pred_qda_best$class))) + geom_point()+ labs(x = "Absolute
Magnitude", y = "Minimum Orbit Intersection", color = "Hazardous", title
= "LDA") + theme(legend.position =
c(0.8, 0.8))

g<- ggplot(test, aes(x = Absolute_Magnitude, y =
Minimum_Orbit_Intersection, color = as.factor(pred_qda_best$class))) +
geom_point()+ labs(x = "Absolute Magnitude", y = "Minimum Orbit
Intersection", color = "Hazardous", title = "QDA") + theme(legend.position = c(0.8, 0.8))

h<- ggplot(test, aes(x = Absolute_Magnitude, y =
Minimum_Orbit_Intersection, color = as.factor(ifelse(pred_ridge_best > 0.5, 1, 0)))) +
geom_point()+ labs(x = "Absolute Magnitude", y = "Minimum Orbit
Intersection", color = "Hazardous", title = "Ridge") + theme(legend.position = c(0.8,
0.8))

i<- ggplot(test, aes(x = Absolute_Magnitude, y =
Minimum_Orbit_Intersection, color = as.factor(ifelse(pred_lasso_best > 0.5, 1, 0)))) +
geom_point()+ labs(x = "Absolute Magnitude", y = "Minimum Orbit
Intersection", color = "Hazardous", title = "Lasso") + theme(legend.position = c(0.8,
0.8))

grid.arrange(e,f,g, h, i, nrow = 2)
```