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 to the Earth. 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 present in the proposed dataset.

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. We proceed with the importation of the data.

```
# 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)
```

Data Uploading

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

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.
```

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

head(nasa_orig,5)

# 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

The first 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.

```
anyNA(nasa_orig)
```

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

We consider now the features described in the dataset that represent the characteristics for each unit. The meaning of each of them is reported 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

Variable	Description
Orbit ID	Orbit ID of the object
Orbit Determination Date	Date when the orbit was defined
Orbit Uncertainity	Uncertainty 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
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 show the same value for all the units in the dataset.

```
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. In particular we want that all of them is codified in numeric type, in order to be able to 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))
```

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 tecniques 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 each model we will compare their performances on the original dataset and on the balanced one.

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. The proportions defined are 0.75 and 0.25.

```
# 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 complete dataset:

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

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

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

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

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

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

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

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

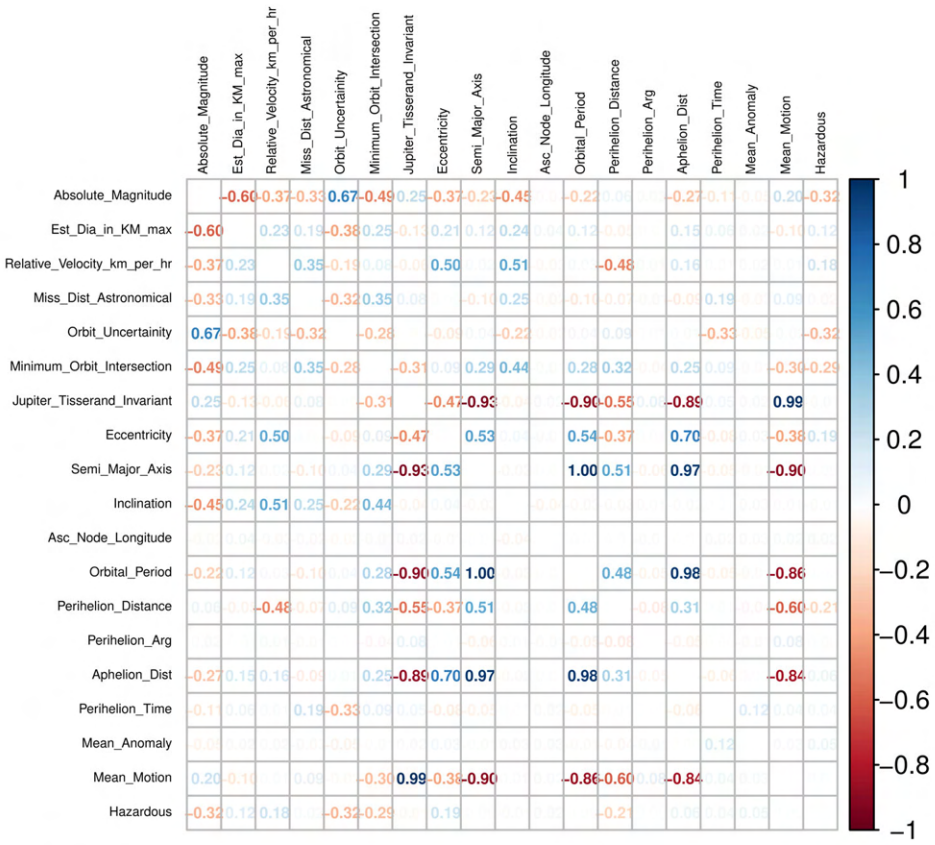
Data Analysis

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.

Correlations

We start looking at the correlation between all the variables for having a general idea: we can observe them in the Figure 1.

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

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



Focusing on the relations with the response , from this image we can observe that the most correlated variable with *Hazardous* are :

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

With this values, we have a starting point for begin to better investigate the relationships between Hazardous and the identified variables. We can observe, for example, the distribution of them, differentiating it for objects classified as dangerous and non-dangerous. We report here the related density plots:

```
# For these plots we consider Hazardous as a factor
Haz<- as.factor(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 Uncertainty
```

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

# 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")


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

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 Uncertainty*. 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), traslations 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 uncertainty 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 again on the correlations matrix shown in Figure 1, we have to note that we have several 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.

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.

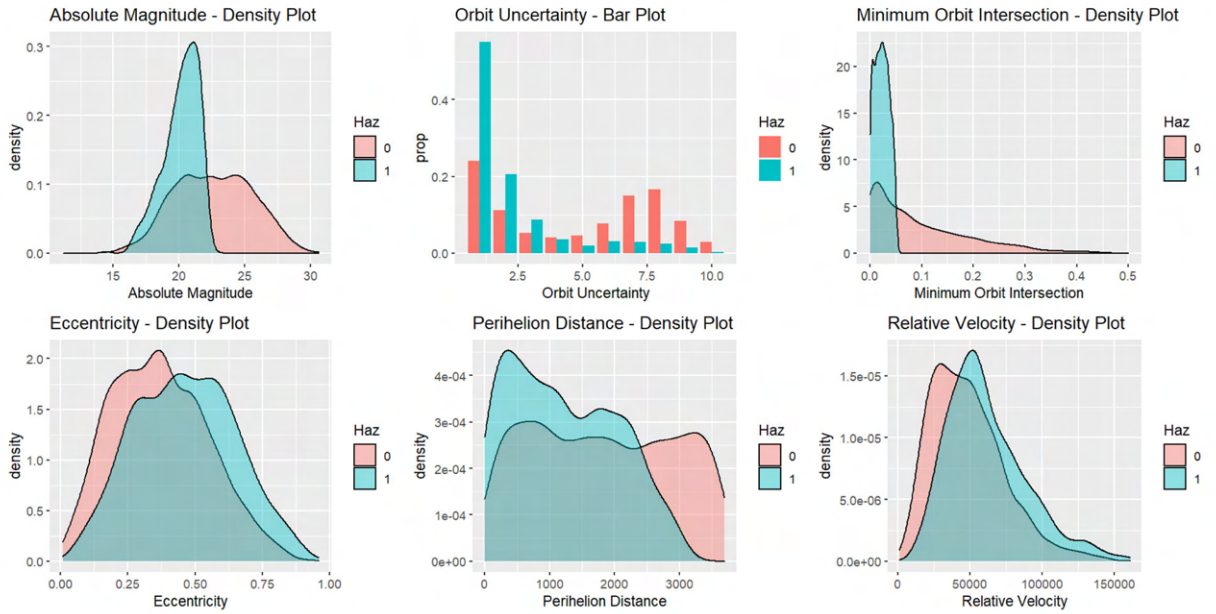


Figure 1: Density plots for NEOs hazardous and not hazardous

```
# We use the "correlation" function from the "correlation" package, where we can specify the computatio

correlation(train, partial = TRUE)

# (The output is not shown for its dimensions).
```

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 Uncertainty	Absolute Magnitude	0.67
Estimated Diameter KM max	Absolute MAgnitude	-0.6

Then, starting from the meanings of the variables we can try to understand why such variables are so correlated.

- 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:

- Then we have also that the value of the *Orbit Uncertainty* 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.

All this existent connections, shown in the next figure, 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.

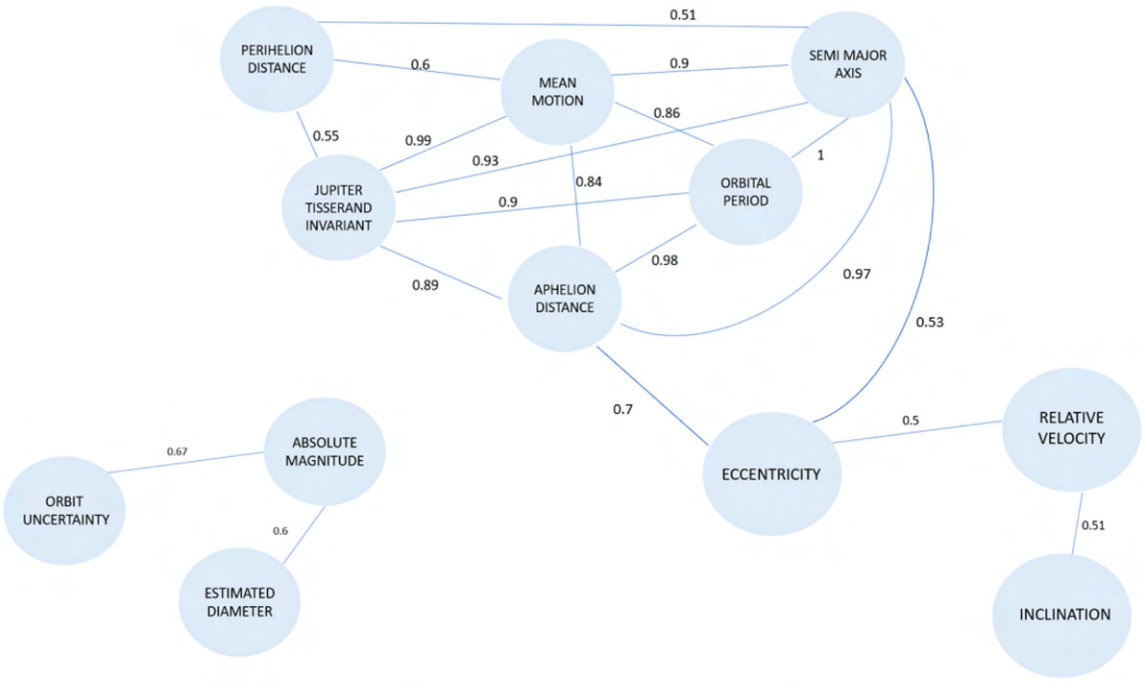


Figure 2: Relations Map for Features

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 *Stepwize Selection* approach for including and exluding iteratively the covariates inside the model. Then, for each built structure we will consider different tresholds 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 to have a less values for *Hazardous* NEOs classified as *Not Hazardous*, maintaing at the same time not too high the quantity of the *False Positives*.

```
library(ROSE)
library(caret)
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 tresholds 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 should delete *Absolute Magnitude*. 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. Reasoning on the true meaning of the correctness of a statistical model, we decide for deleting the covariate in order to remove the collinearity: so all the features considered in the model carry information regarding their relationship to the answer, without considering spurious correlations.

```
# 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
```

```
# 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
```

```
glm_compl<- glm(data = train,
            Hazardous ~.-Aphelion_Dist-Semi_Major_Axis-
              Jupiter_Tisserand_Invariant-
              Eccentricity-Mean_Motion-Absolute_Magnitude,
            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 -
##     Absolute_Magnitude, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7108  -0.2940  -0.0757  -0.0015   3.3061
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  6.548e+02  2.099e+02   3.120  0.00181 **
## Est_Dia_in_KM_max            2.589e-01  5.485e-02   4.719 2.36e-06 ***
## Relative_Velocity_km_per_hr  1.188e-06  3.809e-06   0.312  0.75512
## Miss_Dist_Astronomical       1.015e+00  5.203e-01   1.950  0.05121 .
## Orbit_Uncertainity          -6.766e-01  3.295e-02 -20.533  < 2e-16 ***
## Minimum_Orbit_Intersection  -6.686e+01  3.727e+00 -17.941  < 2e-16 ***
## Inclination                  7.147e-02  8.725e-03   8.191 2.60e-16 ***
## Asc_Node_Longitude           1.901e-04  6.647e-04   0.286  0.77483
## Orbital_Period               2.730e-03  2.733e-04   9.988  < 2e-16 ***
## Perihelion_Distance         -6.315e-04  1.077e-04  -5.865 4.48e-09 ***
## Perihelion_Arg              -1.139e-03  6.814e-04  -1.671  0.09464 .
## Perihelion_Time             -2.659e-04  8.539e-05  -3.114  0.00185 **
## Mean_Anomaly                 1.192e-03  6.759e-04   1.763  0.07785 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3066.2  on 3514  degrees of freedom
## Residual deviance: 1390.2  on 3502  degrees of freedom
## AIC: 1416.2
##
## Number of Fisher Scoring iterations: 8
```

```
# 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 treshold:

pred_glm_compl_04<- ifelse(pred_glm_compl > threshold4, 1, 0)
pred_glm_compl_05<- ifelse(pred_glm_compl > threshold5, 1, 0)
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 that we are including in the model all the covariates inside the dataset. Then we move our attention on the estimated coefficient for each of them: the significative attributes seem to be:

- Estimated Diameter
- Minimum Orbit Intersection
- Orbit Uncertainty
- Inclination
- Orbital Period
- Perihelion Distance
- Perihelion Time

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

```
# Confusion matrix with treshold = 0.4

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

# Confusion matrix with treshold = 0.5

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

# Confusion matrix with treshold = 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	921	51	972
1	46	154	200
Total	967	205	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	939	33	972
1	62	138	200
Total	1001	171	1172

Treshold: 0.6			
	Prediction		
Real Values	0	1	Total
0	950	22	972
1	77	123	200
Total	1027	145	1172

As we can note, increasing the treshold 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 of iteratively adding and/or removing 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 lowers prediction error. There are three strategies of stepwise that we can apply:

- *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-Absolute_Magnitude,
            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 ~ Est_Dia_in_KM_max + Miss_Dist_Astronomical +
##     Orbit_Uncertainity + Minimum_Orbit_Intersection + Inclination +
```

```
##     Orbital_Period + Perihelion_Distance + Perihelion_Arg + Perihelion_Time +
##     Mean_Anomaly, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7123  -0.2927  -0.0763  -0.0015   3.3065
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  6.539e+02  2.097e+02   3.118  0.00182 **
## Est_Dia_in_KM_max            2.597e-01  5.474e-02   4.746 2.08e-06 ***
## Miss_Dist_Astronomical       1.072e+00  4.803e-01   2.231  0.02566 *
## Orbit_Uncertainity          -6.770e-01  3.295e-02 -20.543  < 2e-16 ***
## Minimum_Orbit_Intersection -6.695e+01  3.721e+00 -17.991  < 2e-16 ***
## Inclination                  7.277e-02  7.700e-03   9.451  < 2e-16 ***
## Orbital_Period               2.770e-03  2.367e-04  11.705  < 2e-16 ***
## Perihelion_Distance         -6.505e-04  8.837e-05  -7.361 1.83e-13 ***
## Perihelion_Arg              -1.150e-03  6.810e-04  -1.688  0.09138 .
## Perihelion_Time             -2.655e-04  8.533e-05  -3.111  0.00186 **
## Mean_Anomaly                 1.189e-03  6.757e-04   1.760  0.07838 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3066.2  on 3514  degrees of freedom
## Residual deviance: 1390.4  on 3504  degrees of freedom
## AIC: 1412.4
##
## Number of Fisher Scoring iterations: 8
```

```r
# Computating 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 treshold:

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:

- Estimated Diameter
- Missing Distance
- Minimum Orbit Intersection
- Orbit Uncertainty
- Inclination
- Orbital Period
- Perihelion Distance
- Perihelion Time

As we can note, against the previous results, here the model considers as significant one more feature: *Missing Distance*. This is due to the capacity of the *Stepwise* approach to find the best combination of variables, looking at the AIC value deriving from their removal or insertion. It can be possible, in fact, that the model

shows that new attributes become significant after the elimination of a particular variable.

```
# Confusion matrix with treshold = 0.4

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

# Confusion matrix with treshold = 0.5

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

# Confusion matrix with treshold = 0.6

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

Confusion Matrices - GLM with Stepwise

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	922	50	972
1	46	154	200
Total	968	204	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	940	32	972
1	61	139	200
Total	1001	171	1172

Treshold: 0.6			
	Prediction		
Real Values	0	1	Total
0	950	22	972
1	73	127	200
Total	1023	149	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, so we can conclude that the integration of *Missing Distance* doesn't give so much contribution to the correctness of predictions. This is confirmed by the correlation between the mentioned feature and the response *Hazardous* that is very low (~0.02).

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.

```
# 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-
              Jupiter_Tisserand_Invariant-Eccentricity-
              Absolute_Magnitude-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 - Absolute_Magnitude -
##     Mean_Motion, family = "binomial", data = train_balanced)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5127  -0.2743  -0.0001   0.3969   2.8323
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 5.699e+02  1.793e+02   3.179 0.001478 **
## Est_Dia_in_KM_max           1.812e-01  4.977e-02   3.642 0.000271 ***
## Relative_Velocity_km_per_hr 2.561e-08  3.327e-06   0.008 0.993857
## Miss_Dist_Astronomical      1.202e-02  4.636e-01   0.026 0.979310
## Orbit_Uncertainity         -7.307e-01  2.826e-02 -25.855  < 2e-16 ***
## Minimum_Orbit_Intersection -6.860e+01  3.218e+00 -21.314  < 2e-16 ***
## Inclination                 8.274e-02  7.910e-03  10.460  < 2e-16 ***
```

```
## Asc_Node_Longitude          -3.177e-04  5.861e-04  -0.542 0.587703
## Orbital_Period               3.381e-03  2.559e-04  13.212  < 2e-16 ***
## Perihelion_Distance         -8.891e-04  9.384e-05  -9.475  < 2e-16 ***
## Perihelion_Arg              -2.247e-03  5.750e-04  -3.907 9.33e-05 ***
## Perihelion_Time             -2.304e-04  7.294e-05  -3.158 0.001586 **
## Mean_Anomaly                 1.014e-03  5.666e-04   1.789 0.073615 .
## ---
## 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: 1916.0  on 3502  degrees of freedom
## AIC: 1942
##
## Number of Fisher Scoring iterations: 7
```

```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 treshold:

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)
```

We can immediately note that here we have the presence of more significant variables than before:

- Estimated Diameter
- Minimum Orbit Intersection
- Orbit Uncertainty
- Inclination
- Perihelion Distance
- Orbital Period
- Perihelion Argument
- Perihelion Time

Comparing the resulting significant variables between the *Simple GLM* and the same model applied on the balanced training dataset, we can see that here we have one more variable: *Perihelion Argument*. We observe now, if the introduction of such variable can be an advantage for our classification task.

```r
# Confusion matrix with treshold = 0.4

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

# Confusion matrix with treshold = 0.5

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

# Confusion matrix with treshold = 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	813	159	972
1	16	184	200
Total	829	343	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	829	143	972
1	21	179	200
Total	850	322	1172

Treshold: 0.6			
	Prediction		
Real Values	0	1	Total
0	863	109	972
1	24	176	200
Total	887	285	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 Weigths We can propose another method for balancing the classes: it consists into add different weigths to elements belonging to different classes and in particular we want to valorize examples of the minority one. For that reason, we apply a weigth greater than 1 to the *Hazardous* class.

```
# Definition of the weigths

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

w[train$Hazardous == 1]<- 3

# Model definition:

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

# We compute the reference level R-Squared

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

1/(1-r2)
```

```
## [1] 5.030104
```

```
# 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
##                    8.853946                   5.026370
## Relative_Velocity_km_per_hr     Miss_Dist_Astronomical
##                    2.796999                   1.290893
##          Orbit_Uncertainity  Minimum_Orbit_Intersection
##                    1.662089                   2.654897
## Jupiter_Tisserand_Invariant                Eccentricity
##                2737.189288                  35.502757
##             Semi_Major_Axis                 Inclination
##                2610.196691                   6.318218
```

```
##          Asc_Node_Longitude              Orbital_Period
##                    1.048383                 3080.598672
##         Perihelion_Distance              Perihelion_Arg
##                   43.593305                    1.026152
##               Aphelion_Dist             Perihelion_Time
##                 3361.702975                    1.165945
##                Mean_Anomaly                Mean_Motion
##                    1.065799                 1600.466219
```

```
glm_weighted<- glm(data = train_balanced,
              Hazardous ~.-Aphelion_Dist-Semi_Major_Axis-
              Jupiter_Tisserand_Invariant-Eccentricity-
              Absolute_Magnitude,
              family = "binomial")

# Observation of the model summary:

summary(glm_weighted)
```

```
##
## Call:
## glm(formula = Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis -
##     Jupiter_Tisserand_Invariant - Eccentricity - Absolute_Magnitude,
##     family = "binomial", data = train_balanced)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4814  -0.2788  -0.0001   0.3895   2.7473
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  5.727e+02  1.775e+02   3.226 0.001255 **
## Est_Dia_in_KM_max            1.799e-01  5.041e-02   3.569 0.000358 ***
## Relative_Velocity_km_per_hr -1.504e-06  3.412e-06  -0.441 0.659371
## Miss_Dist_Astronomical       1.199e-02  4.652e-01   0.026 0.979430
## Orbit_Uncertainity          -7.287e-01  2.836e-02 -25.699  < 2e-16 ***
## Minimum_Orbit_Intersection  -6.851e+01  3.216e+00 -21.300  < 2e-16 ***
## Inclination                  8.494e-02  8.011e-03  10.603  < 2e-16 ***
## Asc_Node_Longitude          -3.765e-04  5.870e-04  -0.641 0.521236
## Orbital_Period               2.725e-03  3.782e-04   7.205  5.8e-13 ***
## Perihelion_Distance         -1.004e-03  1.066e-04  -9.415  < 2e-16 ***
## Perihelion_Arg              -2.137e-03  5.755e-04  -3.713 0.000205 ***
## Perihelion_Time             -2.310e-04  7.222e-05  -3.199 0.001381 **
## Mean_Anomaly                 1.041e-03  5.663e-04   1.838 0.066030 .
## Mean_Motion                 -8.372e-01  3.575e-01  -2.342 0.019186 *
## ---
## 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: 1910.5  on 3501  degrees of freedom
## AIC: 1938.5
##
## Number of Fisher Scoring iterations: 7
```

```
# 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 treshold:

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)
```

- Estimated Diameter
- Orbit Uncertainty
- Minimum Orbit Intersection
- Inclination
- Orbital Period
- Perihelion Distance
- Perihelion Argument
- Perihelion Time
- Mean Motion

In this case we consider also the *Mean Motion* attribute. We look now at the confusion matrices in order to understand if this new model could be a good trade-off between the two previous proposed and if the introduction of this new feature can be useful:

```
# Confusion matrix with treshold = 0.4

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

# Confusion matrix with treshold = 0.5

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

# Confusion matrix with treshold = 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	812	160	972
1	16	184	200
Total	828	344	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	834	138	972
1	20	180	200
Total	854	318	1172

Treshold: 0.6			
	Prediction		
Real Values	0	1	Total
0	863	109	972
1	24	176	200
Total	887	285	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. 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.

```r
# 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 ~ Est_Dia_in_KM_max + Orbit_Uncertainity +
##     Minimum_Orbit_Intersection + Inclination + Orbital_Period +
##     Perihelion_Distance + Perihelion_Arg + Perihelion_Time +
##     Mean_Anomaly, family = "binomial", data = train_balanced)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5115  -0.2743  -0.0001   0.3993   2.8351
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  5.710e+02  1.781e+02   3.205 0.001350 **
## Est_Dia_in_KM_max            1.797e-01  4.962e-02   3.622 0.000292 ***
## Orbit_Uncertainity          -7.302e-01  2.811e-02 -25.971  < 2e-16 ***
## Minimum_Orbit_Intersection -6.851e+01  3.205e+00 -21.376  < 2e-16 ***
## Inclination                  8.269e-02  6.885e-03  12.010  < 2e-16 ***
## Orbital_Period               3.387e-03  2.229e-04  15.194  < 2e-16 ***
## Perihelion_Distance         -8.894e-04  7.616e-05 -11.678  < 2e-16 ***
## Perihelion_Arg              -2.217e-03  5.721e-04  -3.875 0.000107 ***
## Perihelion_Time             -2.308e-04  7.248e-05  -3.185 0.001447 **
## Mean_Anomaly                 1.006e-03  5.631e-04   1.787 0.073886 .
## ---
## 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: 1916.3  on 3505  degrees of freedom
## AIC: 1936.3
##
## Number of Fisher Scoring iterations: 7
```

```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 treshold:

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:

- Estimated Diameter
- Minimum Orbit Intersection
- Orbit Uncertainty
- Inclination
- Perihelion Distance
- Perihelion Argument
- Orbital Period
- Perihelion Time

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

```
# Confusion matrix with treshold = 0.4

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

# Confusion matrix with treshold = 0.5

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

# Confusion matrix with treshold = 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	812	160	972
1	16	184	200
Total	828	344	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	829	143	972
1	21	179	200
Total	850	322	1172

Treshold: 0.6			
	Prediction		
Real Values	0	1	Total
0	864	108	972
1	24	176	200
Total	888	284	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.

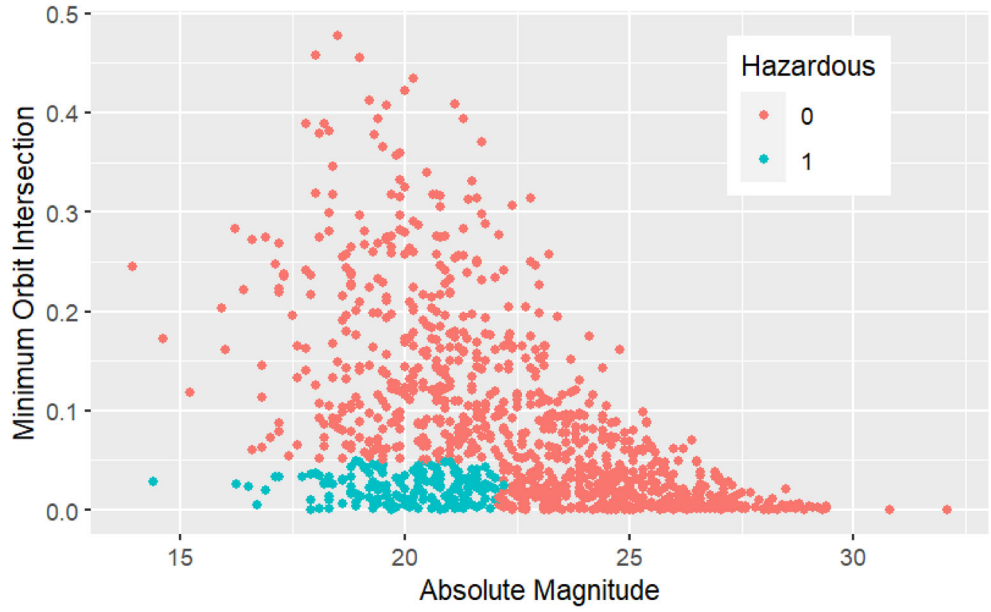
We can compare the ability of classification of the models, looking at how they are able to reproduce the original graphical classification of the points that we can

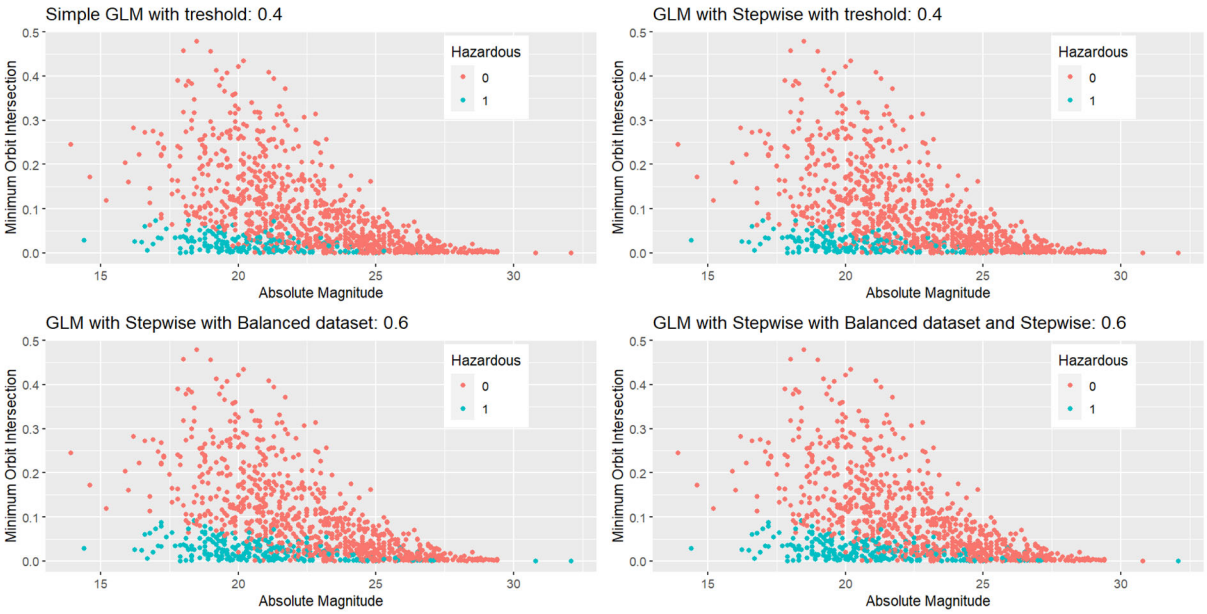
```
# For these plots we consider Hazardous as a factor
Haz_test<- as.factor(test$Hazardous)

# 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 with treshold: 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 with treshold: 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 Stepwise 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 Stepwise with Balanced dataset and Stepwise: 0.6")  +
  theme(legend.position = c(0.8, 0.8))
```

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

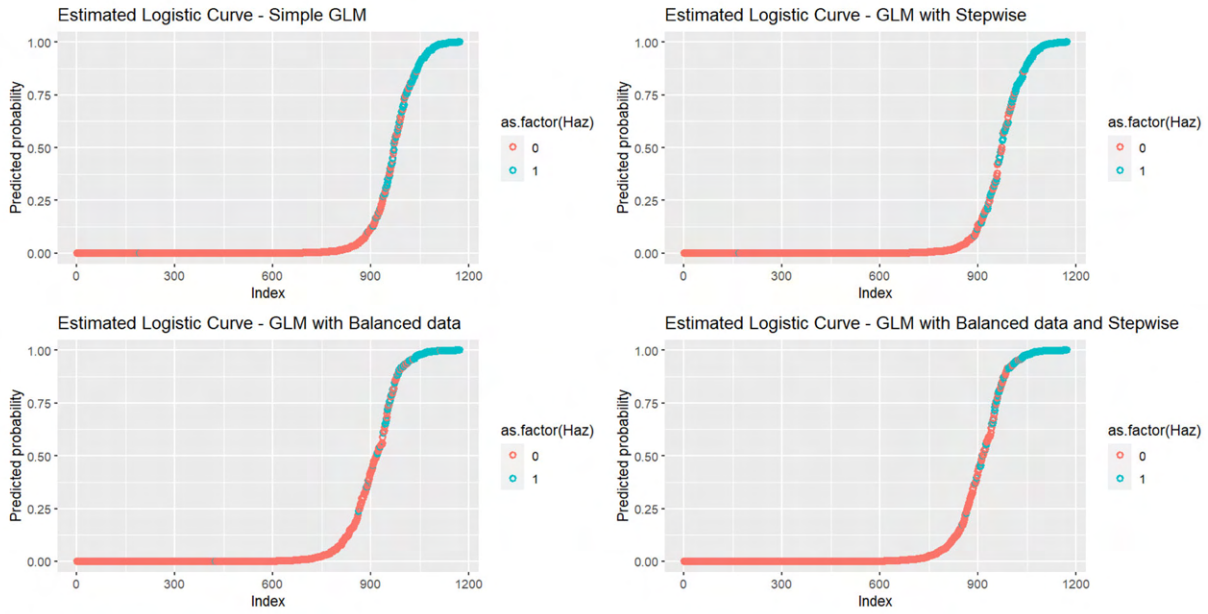




(Commento)

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


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



(Commento)

Linear Discriminant Analysis

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

For this method we have to take in account that we don't have available variables selection techniques: we will start from the models that we obtain considering the *VIF* and on these we will apply the LDA approach. (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.

library(outliers)
```

```
##
## Caricamento pacchetto: 'outliers'
```

```
## Il seguente oggetto è mascherato da 'package:randomForest':
##
##     outlier
```

```
# 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")

# 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
```

```
# Removal of the found outlier:

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

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

```
# 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
```

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

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

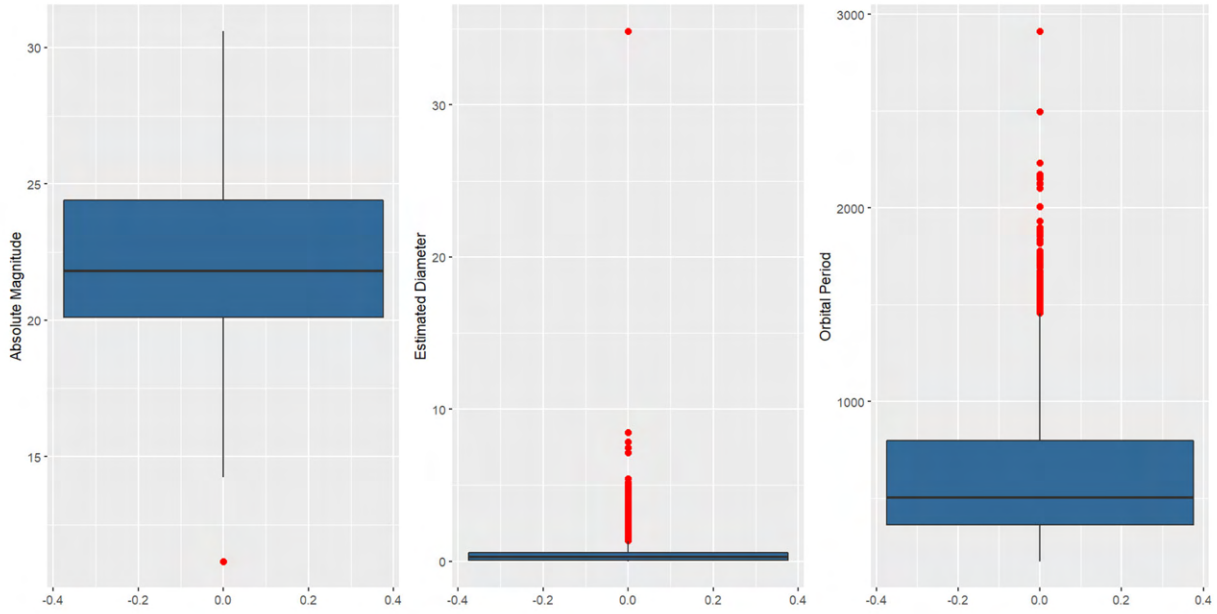
```
g3<- ggplot(data = train, aes(y = Orbital_Period,fill = 2)) +
       geom_boxplot(outlier.colour = "red", outlier.shape = 16,
                    outlier.size = 2)+
       theme(legend.position="none") +
       ylab("Orbital Period")

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
```

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

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

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



Simple LDA

```
library(MASS)

# Model definition:

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

# Observing the model summary:

lda_compl
```

```
## Call:
## lda(Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis - Jupiter_Tisserand_Invariant -
##     Eccentricity - Mean_Motion - Absolute_Magnitude, data = train,
##     family = "binomial")
##
## Prior probabilities of groups:
##         0         1
## 0.8420154 0.1579846
##
## Group means:
##   Est_Dia_in_KM_max Relative_Velocity_km_per_hr Miss_Dist_Astronomical
## 0         0.4096434                    48559.65              0.2566026
## 1         0.7044687                    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
## Est_Dia_in_KM_max            1.136320e-01
## Relative_Velocity_km_per_hr  2.638772e-06
## Miss_Dist_Astronomical       7.355377e-01
## Orbit_Uncertainity          -2.953940e-01
## Minimum_Orbit_Intersection  -1.187238e+01
## Inclination                  1.497487e-02
## Asc_Node_Longitude           2.875093e-04
## Orbital_Period               1.159645e-03
## Perihelion_Distance         -1.555165e-04
## Perihelion_Arg              -2.292307e-04
## Perihelion_Time             -1.395576e-04
## Mean_Anomaly                 5.549170e-04
```

```
# Computing predictions:

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

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

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))
```

```
# Confusion matrix with treshold = 0.4

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

# Confusion matrix with treshold = 0.5

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

# Confusion matrix with treshold = 0.6

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

As we can see, with the same tresholds used until now, the results obtained are worst. 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 treshold = 0.2
```

```
table(test$Hazardous, pred_lda_compl_02)
```

```
##    pred_lda_compl_02
##       0   1
##   0 813 159
##   1  25 175
```

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

```
## [1] 0.1569966
```

```
# Confusion matrix with treshold = 0.3

table(test$Hazardous, pred_lda_compl_03)
```

```
##    pred_lda_compl_03
##       0   1
##   0 877  95
##   1  39 161
```

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

```
## [1] 0.1143345
```

Confusion Matrices - Simple LDA

Treshold: 0.3			
	Predictions		
Real Values	0	1	Total
0	877	95	972
1	39	161	200
Total	916	256	1172

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	913	59	972
1	63	137	200
Total	976	196	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	942	30	972
1	88	112	200
Total	1030	142	1172

(Commento)

The best results is given using 0.3 as treshold and in fact we can see that is the only case where we have that the *False Negative* rate is less than the *False Positive* one.

```
# 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, )
```

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")
```

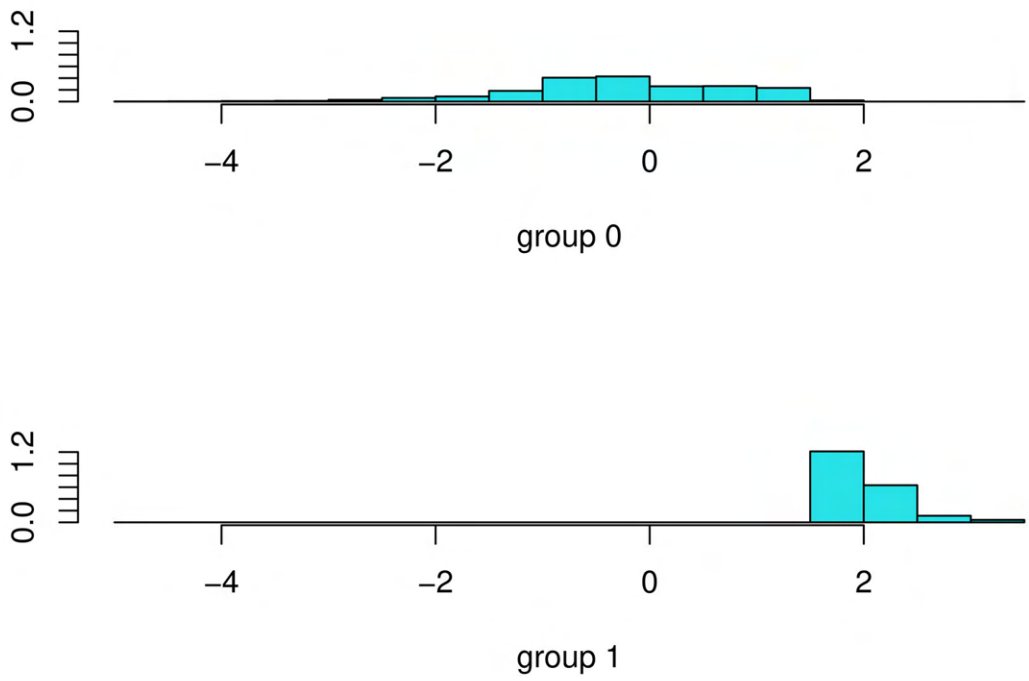


Figure 3: Histograms of how the combinations of variables classify the examples

```r
# 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
```

```r
# 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
```

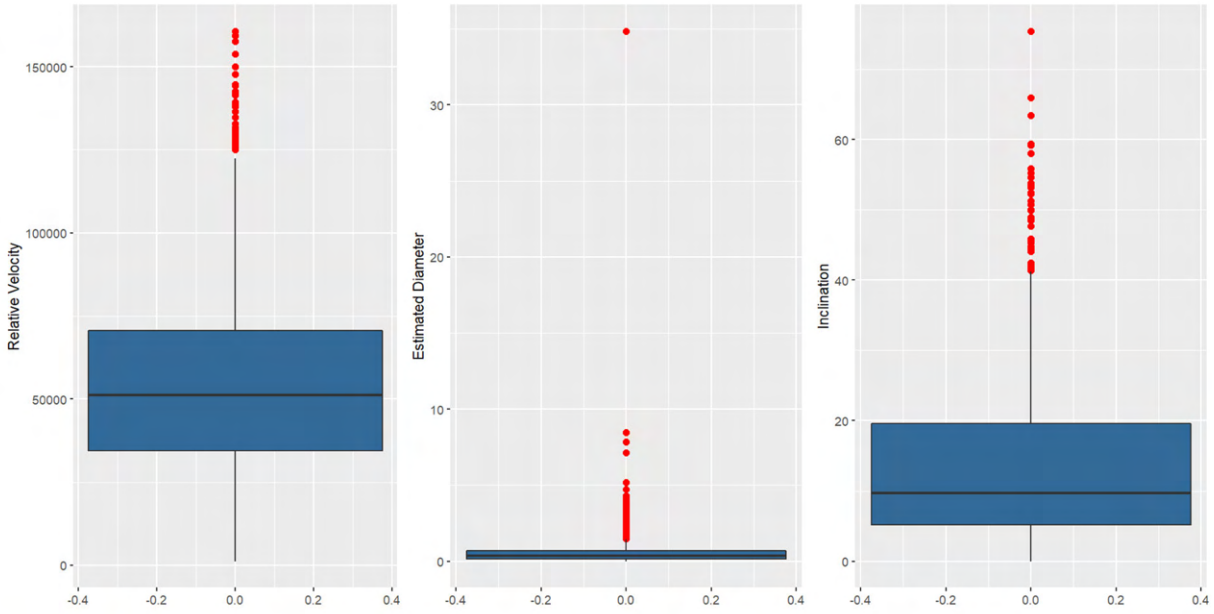
```r
g6<- ggplot(data = train_balanced, aes(y = Inclination,fill = 2)) +
       geom_boxplot(outlier.colour = "red", outlier.shape = 16,
                    outlier.size = 2)+
       theme(legend.position="none") +
       ylab("Inclination")

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
```

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




```
# 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-
              Absolute_Magnitude-Mean_Motion,
              family = "binomial")

# Observing the model summary:

lda_bal
```

```
## Call:
## lda(Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis - Jupiter_Tisserand_Invariant -
##     Eccentricity - Absolute_Magnitude - Mean_Motion, data = train_balanced,
##     family = "binomial")
##
## Prior probabilities of groups:
##         0         1
## 0.5145299 0.4854701
##
## Group means:
##   Est_Dia_in_KM_max Relative_Velocity_km_per_hr Miss_Dist_Astronomical
## 0         0.4271487                    48769.74              0.2577234
## 1         0.6951774                    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
## Est_Dia_in_KM_max            2.612380e-02
## Relative_Velocity_km_per_hr -1.097468e-06
## Miss_Dist_Astronomical       5.708435e-01
## Orbit_Uncertainity          -3.375235e-01
## Minimum_Orbit_Intersection  -1.469383e+01
## Inclination                  2.391515e-02
## Asc_Node_Longitude           6.981788e-05
## Orbital_Period               1.342286e-03
## Perihelion_Distance         -3.054053e-04
## Perihelion_Arg              -9.057266e-04
## Perihelion_Time             -1.143302e-04
## Mean_Anomaly                 4.714965e-04
```

```
# 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 treshold:

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))
```

```
# Confusion matrix with treshold: 0.3

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

# Confusion matrix with treshold: 0.4

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

# Confusion matrix with treshold: 0.5

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

# Confusion matrix with treshold: 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)
```

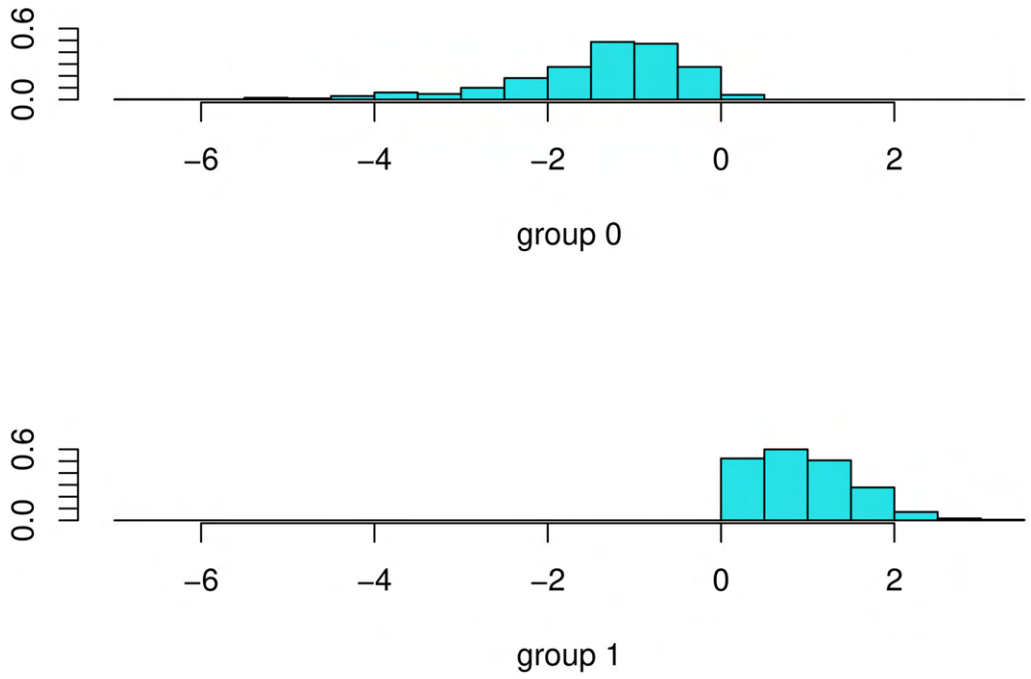


Figure 4: Histograms of how the combinations of variables classify the examples

Confusion Matrices - LDA with Balanced dataset

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	738	234	972
1	16	184	200
Total	754	418	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	784	188	972
1	21	179	200
Total	805	367	1172

Treshold: 0.6			
	Predictions		
Real Values	0	1	Total
0	825	147	972
1	26	174	200
Total	851	321	1172

Here we can see that we have in any case the *False Negative* rate less than the *False Positive* one. The best results, according to the initial considerations, are obtained with the threshold equal to 0.6: the total error is the less one but in the while we have an increased error on the classification of the hazardous element as *Not Hazardous*.

Quadratic Discriminant Analysis

con qda dovremmo essere in grado di osservare dei risultati migliori rispetto alla lda: questo perchè lda tende a produrre risultati migliori con poche osservazioni di training –> importante ridurre la varianza. se l'insieme delle osservazioni è più ampio e di conseguenza la varianza non è rilevante, qda si mostra un metodo più efficace per la produzione di modelli di classificazione.

(Nota sulla variabile da rimuovere perchè non continua)

Simple QDA

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

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

# Observing the model summary:

qda_compl
```

```
## Call:
## qda(Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis - Jupiter_Tisserand_Invariant -
##     Eccentricity - Mean_Motion - Absolute_Magnitude - Orbit_Uncertainity,
##     data = train, family = "binomial")
##
## Prior probabilities of groups:
##         0         1
## 0.8420154 0.1579846
##
## Group means:
##   Est_Dia_in_KM_max Relative_Velocity_km_per_hr Miss_Dist_Astronomical
## 0         0.4096434                    48559.65              0.2566026
## 1         0.7044687                    61828.79              0.2651441
##   Minimum_Orbit_Intersection Inclination Asc_Node_Longitude Orbital_Period
## 0                 0.09481278    13.53324           169.9012       632.8233
## 1                 0.02291273    13.15031           176.2912       637.5504
##   Perihelion_Distance Perihelion_Arg Perihelion_Time Mean_Anomaly
## 0            1821.232       185.2189         2457714     178.9938
## 1            1222.000       185.2448         2457815     194.5807
```

```r
# Computing predictions:

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


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

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 treshold: 0.3

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

# Confusion matrix with treshold: 0.4

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

# Confusion matrix with treshold: 0.5

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

# Confusion matrix with treshold: 0.6

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

Confusion Matrices - Simple QDA

Treshold: 0.3			
	Predictions		
Real Values	0	1	Total
0	805	167	972
1	22	178	200
Total	827	345	1172

Treshold: 0.4			
	Predictions		
Real Values	0	1	Total
0	869	103	972
1	49	151	200
Total	918	254	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	903	69	972
1	63	137	200
Total	966	206	1172

Considering high treshold as for example 0.6, we can see oh the rate of *False Positive* becomes less than the *False Negatives*, so in order to compare the best results for the *Simple QDA* model we consider less tresholds, starting also here from 0.3 up to 0.5 .

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-
              Absolute_Magnitude-Mean_Motion- Orbit_Uncertainity,
              family = "binomial")
```

```r
# Observing the model summary:

qda_bal
```

```
## Call:
## qda(Hazardous ~ . - Aphelion_Dist - Semi_Major_Axis - Jupiter_Tisserand_Invariant -
##     Eccentricity - Absolute_Magnitude - Mean_Motion - Orbit_Uncertainity,
##     data = train_balanced, family = "binomial")
##
## Prior probabilities of groups:
##         0         1
## 0.5145299 0.4854701
##
## Group means:
##   Est_Dia_in_KM_max Relative_Velocity_km_per_hr Miss_Dist_Astronomical
## 0         0.4271487                   48769.74              0.2577234
## 1         0.6951774                   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 treshold:

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 treshold: 0.3

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

# Confusion matrix with treshold: 0.4

table(test$Hazardous, pred_qda_bal_04)
mean(pred_qda_bal_04!=test$Hazardous)

# Confusion matrix with treshold: 0.5

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

# Confusion matrix with treshold: 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	608	364	972
1	5	195	200
Total	613	559	1172

Treshold: 0.5			
	Predictions		
Real Values	0	1	Total
0	661	311	972
1	8	192	200
Total	669	503	1172

Treshold: 0.6			
	Predictions		
Real Values	0	1	Total
0	732	240	972
1	15	185	200
Total	747	425	1172

Regularized Regression

Another approach that we can try is to avoid *Multicollinearity* using *Regularitazion* methods. In fact, the presence of multicollinearity is reflected the estimators will tend to have very large variants, although with small bias. However, estimators that have very large variants will produce poor estimates. This phenomenon is referred to as *Overfitting*. In order to avoid it, in this section we fit generalized models adding 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.

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])
haz_train_ridge<- (train_balanced$Hazardous*2)-1
haz_test_ridge<- (test$Hazardous * 2)-1

# Model definition:

ridge<- glmnet(y = as.factor(haz_train_ridge),
               x = train_bal_mat, alpha = 0, family = "binomial")

# We look for the optimum value for Lambda parameter

pred_ridge <- predict(ridge, newx = test_mat)

# Computing prediction with different values of Lambda:

errors <- (as.numeric(haz_test_ridge) - pred_ridge)^2 %>% apply(2, mean)

# Computing the errors done with each value of lambda in order to find the less one:
lambda_opt_ind <- errors %>% which.min()
lambda_opt_ridge <- ridge$lambda[lambda_opt_ind]

pred_ridge <- pred_ridge[, lambda_opt_ind]

pred_ridge_class<- predict(ridge, test_mat, type = "class", s = lambda_opt_ridge)
table(haz_test_ridge, pred_ridge_class)
```

Lasso Regression

```
# Model definition:

lasso<- glmnet(y = as.factor(haz_train_ridge),
               x = train_bal_mat, alpha = 1, family = "binomial")

# We look for the optimum value for Lambda parameter

pred_lasso <- predict(lasso, newx = test_mat)

# Computing prediction with different values of Lambda:

errors <- (as.numeric(haz_test_ridge) - pred_lasso)^2 %>% apply(2, mean)

lambda_opt_ind <- errors %>% which.min()
lambda_opt_lasso <- lasso$lambda[lambda_opt_ind]

pred_lasso <- pred_lasso[, lambda_opt_ind]
pred_lasso_class<- predict(lasso, test_mat, type = "class", s = lambda_opt_lasso)

table(haz_test_ridge, pred_lasso_class)
```

Ridge Regression			
	Predictions		
Real Values	-1	1	Total
-1	835	137	972
1	22	178	200
Total	857	315	1172

Lasso Regression			
	Predictions		
Real Values	-1	1	Total
-1	865	107	972
1	7	193	200
Total	872	300	1172

Final Considerations

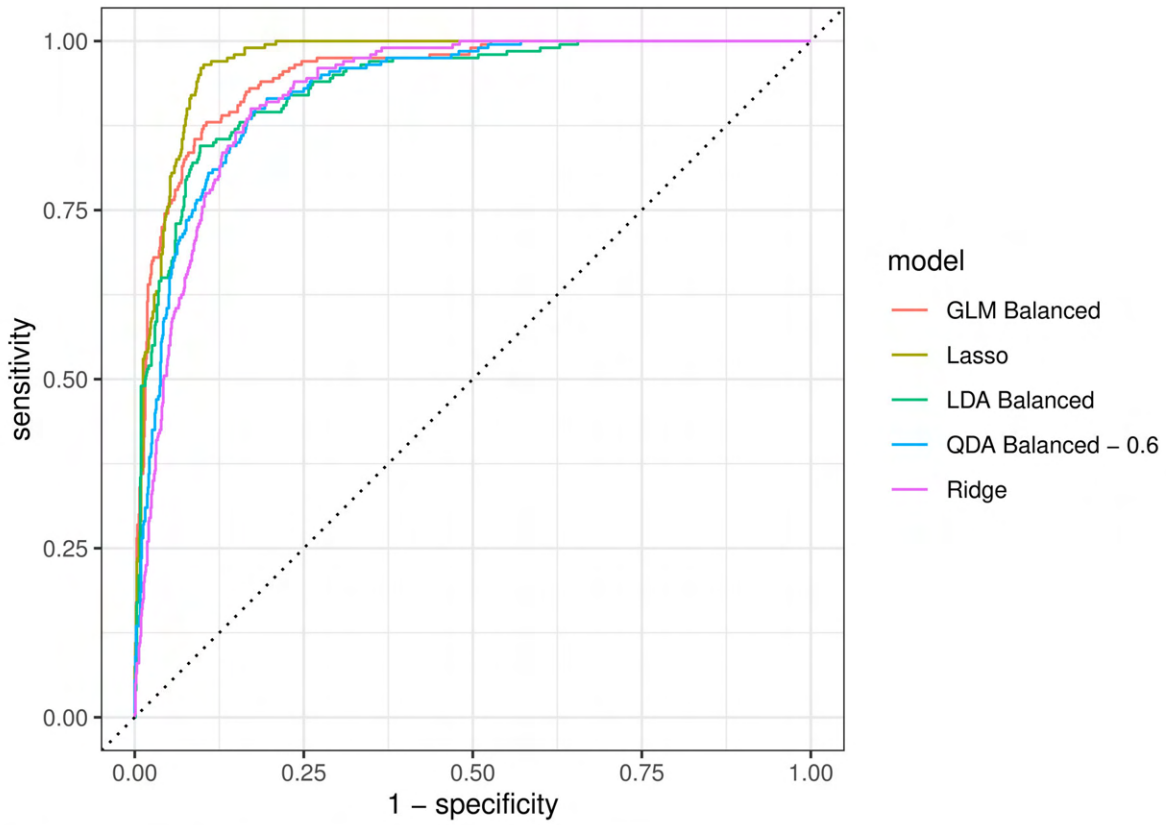
We can now summarize the obtained results with all the types of models we have implemented, choosing particular metrics in order to compare them and their performances on the same level. Referring to the nature of the presented problem of classification of hazardous NEOs for the Earth, we decide to consider 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*.

Comparison of Models

Model	Overall Error Rate (FalsePositive + FalseNegative)/(Total)	False Negative Rate FalseNegative/(FalseNegative+TruePositive)
Simple GLM - 0.4	0.083	0.230
Simple GLM - 0.5	0.081	0.310
Simple GLM - 0.6	0.084	0.385
GLM with Stepwise - 0.4	0.082	0.230
GLM with Stepwise - 0.5	0.079	0.305
GLM with Stepwise - 0.6	0.081	0.365
GLM with Balanced dataset - 0.4	0.149	0.080
GLM with Balanced dataset - 0.5	0.140	0.105
GLM with Balanced dataset - 0.6	0.113	0.120
GLM with Balanced dataset and Stepwise - 0.4	0.150	0.080
GLM with Balanced dataset and Stepwise - 0.5	0.140	0.105
GLM with Balanced dataset and Stepwise - 0.6	0.113	0.120
Simple LDA - 0.3	0.114	0.195
Simple LDA - 0.4	0.104	0.315
Simple LDA - 0.5	0.101	0.440
LDA with Balanced dataset - 0.3	0.213	0.080
LDA with Balanced dataset - 0.4	0.178	0.105
LDA with Balanced dataset - 0.5	0.148	0.130
Simple QDA - 0.3	0.161	0.110
Simple QDA - 0.4	0.130	0.245
Simple QDA - 0.5	0.113	0.315
QDA with Balanced dataset - 0.4	0.315	0.025
QDA with Balanced dataset - 0.5	0.272	0.040
QDA with Balanced dataset - 0.6	0.217	0.075
Ridge Regression - 0.55	0.136	0.110
Lasso Regression - 0.15	0.097	0.035

Curva ROC

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

prediction <- tibble(truth = as.factor(test$Hazardous))
prediction <- prediction %>% mutate(pred = as.numeric(pred_glm_bal))%>%mutate(model= "GLM Balanced")%>%
  add_row(truth = as.factor(test$Hazardous), pred = post_lda_bal[,2], model= "LDA Balanced")%>%
  add_row(truth = as.factor(test$Hazardous), pred = post_qda_bal[,2], model= "QDA Balanced - 0.6")%>%
  add_row(truth = as.factor(test$Hazardous), pred = pred_ridge, model= "Ridge")%>%
  add_row(truth = as.factor(test$Hazardous), pred = pred_lasso, model= "Lasso")


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




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

```
## Area under the curve: 0.9508
```

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

```
## Area under the curve: 0.9343
```

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

```
## Area under the curve: 0.9271
```

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

```
## Area under the curve: 0.9238
```

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

```
## Area under the curve: 0.9683
```