Case Study: JPL Small Body Classification in R

Veneet Bhardwaj

05/01/2021

Contents

1 Introduction

A NEO, Near Earth Object, is a small Solar System body (an asteroid or comet), which comes in proximity with Earth during its orbit.

A PHO, Potentially Hazardous Object, is a NEO with an orbit that intersects that of Earth and hence there is a probability of a collision.

When a NEO is detected, like all other small Solar System bodies, its positions and brightness are submitted to the International Astronomical Union (IAU) Minor Planet Center (MPC) for cataloging. The MPC maintains separate lists of confirmed NEOs and potential NEOs.

NEOs are also cataloged by two separate units of the Jet Propulsion Laboratory (JPL) of the National Aeronautics and Space Administration (NASA): the Center for Near Earth Object Studies (CNEOS) and the Solar System Dynamics Group.

This case study is on making a classification model using R, with the JPL dataset.

1.1 Objective

→ The **Objective** of this case study to build a machine learning model to predict the classification of the small solar bodies into PHO, NEO and SO (other small solar bodies).

Target measures :

- Recall for each category above 0.99
- F1 Score for each category above 0.99

The dataset used is the data available at :

https://ssd.jpl.nasa.gov/sbdb_query.cgi

_ Downloaded file : Nasa_JPL_SmallBody.csv ~555Mb in size _

_ Compressed split files have been uploaded in the folder https://raw.githubusercontent.com/VeneetBhardwaj/R/ _

1.2 A Brief History

source wikipedia

Comets were the first near-Earth objects to be observed by us.

Their extraterrestrial nature was recognized and confirmed only after Tycho Brahe tried to measure the distance of a comet through its parallax in 1577 and the lower limit he obtained was well above the Earth diameter.

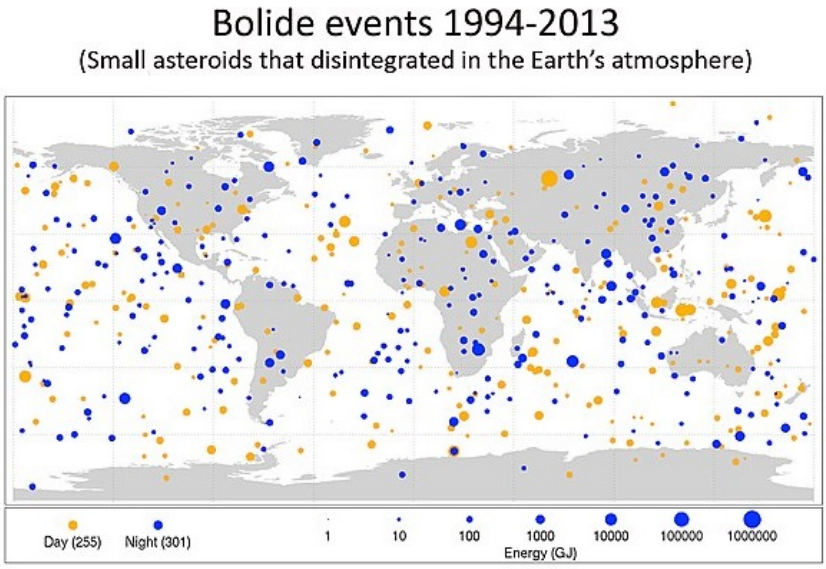
- In 1705, Edmond Halley published his orbit calculations of a NEO, the Halley Comet. The 1758–1759 return of Halley Comet was the first comet appearance that had been predicted.
- It has been said that Lexell comet of 1770 was the first discovered Near-Earth object.
- The first near-Earth asteroid to be discovered was 433 Eros in 1898.
- As of January 2019, five near-Earth comets and five near-Earth asteroids have been visited by spacecraft. A small sample of one NEO was returned to Earth in 2010, and similar missions are in progress.

The first astronomical program dedicated to the discovery of near-Earth asteroids was the Palomar Planet-Crossing Asteroid Survey, started in 1973 by astronomers Eugene Shoemaker and Eleanor Helin.

Several surveys have undertaken the NEO detection.

- Lincoln Near-Earth Asteroid Research (LINEAR)
- Spacewatch, Near-Earth Asteroid Tracking (NEAT)
- Lowell Observatory Near-Earth-Object Search (LONEOS)
- Catalina Sky Survey (CSS)
- Campo Imperatore Near-Earth Object Survey (CINEOS)
- Japanese Spaceguard Association
- Asiago-DLR Asteroid Survey (ADAS)
- Near-Earth Object WISE (NEOWISE).

In 2005, the original USA Spaceguard mandate to detect 90% of near-earth asteroids over 1 km (0.62 mi) diameter, was extended by the George E. Brown, Jr. Near-Earth Object Survey Act, which calls for NASA to detect 90% of NEOs with diameters of 140 m (460 ft) or greater, by 2020.



As of January, 2020, it is estimated that less than half of these have been found. NASA maintains an automated system to evaluate the threat from known NEOs over the next 100 years, which generates the continuously updated Sentry Risk Table.

2 Analysis

```
#load the required libraries
library(knitr)
library(markdown)
options(digits = 5)
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
if(!require(Hmisc)) install.packages("Hmisc", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")
if(!require(nnet)) install.packages("nnet", repos = "http://cran.us.r-project.org")
if(!require(naivebayes)) install.packages("naivebayes", repos = "http://cran.us.r-project.org")
if(!require(ranger)) install.packages("ranger", repos = "http://cran.us.r-project.org")
if(!require(gbm)) install.packages("gbm", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(wsrf)) install.packages("wsrf", repos = "http://cran.us.r-project.org")
```

2.1 Data setup

The JPL small body data set is downloaded from **https://ssd.jpl.nasa.gov/sbdb_query.cgi**

The downloaded file is Nasa_JPL_SmallBody.csv (~555Mb)

```
data <- data.table::fread("Nasa_JPL_SmallBody.csv", na.strings = c(NA,"NA",""))
dim(data)
```

```
## [1] 1081059      59
```

There are 1081059 observations (rows) across 59 columns in the dataset.

```
names(data)
```

```
##  [1] "id"             "full_name"      "pdes"           "neo"
##  [5] "pha"            "H"              "G"              "diameter"
##  [9] "extent"         "albedo"         "rot_per"        "orbit_id"
## [13] "epoch"          "epoch_mjd"      "epoch_cal"      "equinox"
## [17] "e"              "a"              "q"              "i"
## [21] "om"             "w"              "ma"             "ad"
## [25] "n"              "tp"             "tp_cal"         "per"
## [29] "per_y"          "moid"           "moid_ld"        "moid_jup"
## [33] "t_jup"          "sigma_e"        "sigma_a"        "sigma_q"
## [37] "sigma_i"        "sigma_om"       "sigma_w"        "sigma_ma"
## [41] "sigma_ad"       "sigma_n"        "sigma_tp"       "sigma_per"
## [45] "class"          "producer"       "data_arc"       "first_obs"
## [49] "last_obs"       "n_obs_used"     "n_del_obs_used" "n_dop_obs_used"
## [53] "condition_code" "rms"            "two_body"       "A1"
## [57] "A2"             "A3"             "DT"
```

Brief description on the columns of the data set are shown below. These have been *collated from the* ***https://ssd.jpl.nasa.gov*** *website.*

```
#load the descriptions csv.
data$id <- str_trim(data$id)
desc_d <- as.data.frame(data.table::fread("JPLdesc.csv"))
desc_d[,2:4] %>% knitr::kable()
```

column_name	Description	Units
id	Object Internal Database Id	
full_name	Object Full Name/Designation	
pdes	Object Primary Designation	
neo	Near-Earth Object (Neo) Flag	
pha	Potentially Hazardous Asteroid (Pha) Flag	
H	Absolute Magnitude Parameter	
G	Magnitude Slope Parameter (Default Is 0.15)	
diameter	Object Diameter (From Equivalent Sphere)	km
extent	Object Bi/Tri-Axial Ellipsoid Dimensions	km
albedo	Geometric Albedo	
rot_per	Rotation Period	h
orbit_id	Orbit Solution Id	
epoch	Epoch Of Osculation In Julian Day Form	TDB
epoch_mjd	Epoch Of Osculation In Modified Julian Day Form	TDB
epoch_cal	Epoch Of Osculation In Calendar Date/Time Form	TDB
equinox	Equinox Of Reference Frame	
e	Eccentricity	
a	Semi-Major Axis	au
q	Perihelion Distance	au
i	Inclination; Angle With Respect To X-Y Ecliptic Plane	deg
om	Longitude Of The Ascending Node	deg
w	Argument Of Perihelion	deg
ma	Mean Anomaly	deg
ad	Aphelion Distance	au
n	Mean Motion	deg/d
tp	Time Of Perihelion Passage	TDB
tp_cal	Time Of Perihelion Passage	ET
per	Sidereal Orbital Period	d
per_y	Sidereal Orbital Period	years
moid	Earth Minimum Orbit Intersection Distance	au
moid_ld	Earth Minimum Orbit Intersection Distance	LD
moid_jup	Jupiter Minimum Orbit Intersection Distance	au
t_jup	Jupiter Tisserand Invariant	
sigma_e	Eccentricity (1-Sigma Uncertainty)	au
sigma_a	Semi-Major Axis (1-Sigma Uncertainty)	au
sigma_q	Perihelion Distance (1-Sigma Uncertainty)	deg
sigma_i	Inclination (1-Sigma Uncertainty)	deg
sigma_om	Long. Of The Asc. Node (1-Sigma Uncertainty)	deg
sigma_w	Argument Of Perihelion (1-Sigma Uncertainty)	deg
sigma_ma	Mean Anomaly (1-Sigma Uncertainty)	deg
sigma_ad	Aphelion Distance (1-Sigma Uncertainty)	au
sigma_n	Mean Motion (1-Sigma Uncertainty)	deg/d
sigma_tp	Time Of Peri. Passage (1-Sigma Uncertainty)	d
sigma_per	Sidereal Orbital Period (1-Sigma Uncertainty)	d

column_name	Description	Units
class	Orbit Classification	
producer	Name Of Person (Or Institution) Who Computed The Orbit	
data_arc	Number Of Days Spanned By The Data-Arc	d
first_obs	Date Of First Observation Used In The Orbit Fit	UT
last_obs	Date Of Last Observation Used In The Orbit Fit	UT
n_obs_used	Number Of Observations (All Types) Used In Fit	
n_del_obs_used	Number Of Delay-Radar Observations Used In Fit	
n_dop_obs_used	Number Of Doppler-Radar Observations Used In Fit	
condition_code	Orbit Condition Code (Mpc 'U' Parameter)	
rms	Normalized Rms Of Orbit Fit	arcsec
two_body	2-Body Dynamics Used Flag	T/F
A1	Non-Grav. Radial Parameter	
A2	Non-Grav. Transverse Parameter	
A3	Non-Grav. Normal Parameter	
DT	Non-Grav. Peri.-Maximum Offset	d

Observations on the variables (columns) property in the dataset:

- The intent of this case study is to classify the small body objects into the appropriate flags of neo Near-Earth Object (Neo) Flag and pha Potentially Hazardous Asteroid (Pha) Flag.
- Some columns are descriptive of the object like *Object full name* and *name of the person who computed the orbit.*
- Some are regarding objects observation properties, like the *number of observations used for orbit fit*, *date of first and last observation* etc..
- There are also the 1-Sigma Uncertainty of the values in the data set.
- There are variables which are orbital observations and the observations of the object properties. These are the variables which will be used in making the classification model.

2.2 Outcome Variable

The Outcome Variable (to be predicted by the model) needs to be setup from a combination of 2 columns - *pha* and *neo*. Both these have binary outcomes of "Y" and "N".

However there are 12054 missing values in pha and 3539 in neo, and these observations (rows) are being removed.

```
#check for missing values
cat("pha missing values: ", sum(is.na(data$pha)),
    "\nneo missing values : ", sum(is.na(data$neo)))
```

```
## pha missing values:  12054
## neo missing values :  3539
```

The *neo_class* **will have three possible values, which are defined as:**

$$PHO \rightarrow Potential\ Hazard$$

$$NEO \rightarrow Near\ Earth\ Object(not\ Potential\ Hazard)$$

$$SO \rightarrow Other\ (Solar)\ Object(not\ Near\ Earth\ object\ \&\ not\ Potential\ Hazard)$$

Setting up the Outcome Variable :

```
#This is the outcome variable
data$pha <- as.character(data$pha)
data$neo <- as.character(data$neo)
#remove the missing values.
data <- data[complete.cases(data$neo),]
data <- data[complete.cases(data$pha),]
#convert character to numeric,
data$pha <- ifelse(data$pha == "Y",1,0)
data$neo <- ifelse(data$neo == "Y",1,0)
#define the classes
data <- data %>%
  mutate(neo_class = as.factor(ifelse(pha == 1,"PHO",
                               ifelse(pha == 0 & neo == 1,"NEO",
                                      ifelse(pha == 0 & pha == 0,"SO","Error")))))
describe(data$neo_class)
```

```
## data$neo_class
##        n  missing distinct
##  1069002        0        3
##
## Value          NEO     PHO      SO
## Frequency    23544    2163 1043295
## Proportion   0.022   0.002   0.976
```

- With only the complete cases of the flags being used, there is no missing data of the outcome variable.
- The proportions of the PHO and the NEO are very low in the dataset, with 97.6% of the objects being classified as SO.

2.3 Missing data

Variables with missing data:

```
#list features with less than 100% data
m_chk <- tibble()
data_f <- data.frame(data)
for(iter in 1:(ncol(data_f))){
  x <- sum(is.na(data_f[,iter]))
  cname <- as.character(colnames(data_f[iter]))
  uni <- as.character(n_distinct(data_f[iter]))
  perc <- round((((dim(data)[1]-x)/(dim(data_f)[1]))*100),1)
  if (x > 0) {
    m_chk <- bind_rows(m_chk,data_frame(Variable=cname,
                                        Distinct=uni,
                                        Missing=as.character(x),
                                        percentage=str_c(perc,"%")))}

}
m_chk %>% knitr::kable()
```

Variable	Distinct	Missing	percentage
pdes	547967	521036	51.3%
H	9070	4007	99.6%
G	48	1068883	0%
diameter	16782	929002	13.1%
extent	19	1068984	0%
albedo	1055	930153	13%
rot_per	12801	1045918	2.2%
ma	1069001	1	100%
moid_jup	213322	1	100%
sigma_e	274855	1	100%
sigma_a	290958	1	100%
sigma_q	272278	1	100%
sigma_i	232282	1	100%
sigma_om	233199	1	100%
sigma_w	277495	1	100%
sigma_ma	285767	1	100%
sigma_ad	286819	1	100%
sigma_n	263808	1	100%
sigma_tp	307291	1	100%
sigma_per	300529	1	100%
data_arc	22611	337	100%
n_del_obs_used	26	1068199	0.1%
n_dop_obs_used	21	1068162	0.1%
condition_code	11	4	100%
rms	63559	1	100%
two_body	1	1069002	0%
A1	1	1069002	0%
A2	1	1069002	0%
A3	1	1069002	0%
DT	1	1069002	0%

- $H, G, diameter, albedo, rot\_per$ are properties of the object, and useful in building a model. But besides H, the remaining variables have a large number of missing values and hence will not be used.

The observations with missing data, for the variables intended to be used for the model, are removed.

```
dim(data)
```

```
## [1] 1069002      60
```

2.4 Model Variables

The original dataset, post cleaning, is now split into two datasets

- The model Variables
- The descriptive dataset (*details descriptive of the objects*)

```
#descriptive and model variables dataset separated
model_variables <- data %>%  select(id,class,neo_class,H,e,t_jup,
                                    a,q,ad,moid,moid_jup,i,om,w,per,n,ma,rms)

descriptive_data_full <- data %>% select(setdiff(names(data),names(model_variables)))
descriptive_data_full <- bind_cols(id=data$id,moid=data$moid,descriptive_data_full)
#data.table::fwrite(descriptive_data,'descriptive_data_full.csv')
descriptive_data <- descriptive_data_full %>%
  select(id,full_name,producer,neo,pha,per_y,moid,first_obs,last_obs)
data.table::fwrite(descriptive_data,'descriptive_data.csv')
#use only the complete cases. drop the remaining.
model_variables <- model_variables[complete.cases(model_variables),]
```

To make the classification model the **model_variables** will be used.

The list of the columns in the dataset are :

```
# description of the features to be used in the model
x <- ifelse(c(desc_d$column_name) %in% c(names(model_variables)),1,0)
m_chk <- tibble()
for(iter in 1:59){  if (x[iter]==1) {
  m_chk <- bind_rows(m_chk,data_frame(Variable=desc_d[iter,2],
                                      Description=desc_d[iter,3],
                                      Units=desc_d[iter,4]))}}
m_chk %>% knitr::kable()
```

Variable	Description	Units
id	Object Internal Database Id	
H	Absolute Magnitude Parameter	
e	Eccentricity	
a	Semi-Major Axis	au
q	Perihelion Distance	au
i	Inclination; Angle With Respect To X-Y Ecliptic Plane	deg
om	Longitude Of The Ascending Node	deg
w	Argument Of Perihelion	deg
ma	Mean Anomaly	deg
ad	Aphelion Distance	au
n	Mean Motion	deg/d
per	Sidereal Orbital Period	d
moid	Earth Minimum Orbit Intersection Distance	au
moid_jup	Jupiter Minimum Orbit Intersection Distance	au
t_jup	Jupiter Tisserand Invariant	
class	Orbit Classification	
rms	Normalized Rms Of Orbit Fit	arcsec

```
dim(model_variables)
```

```
## [1] 1064993      18
```

There are 1064993 observations (rows) across 18 columns (including the Outcome Variable) in the dataset to be used for making the Classification Model.

2.5 Validation and Model datasets

```
# Validation set will be 10% of model_variables data
set.seed(131825431, sample.kind = "Rounding")

validation_index <- createDataPartition(model_variables$neo_class,
                                        times = 1, p = 0.1, list = FALSE)

JPL_data <- model_variables[-validation_index,]
JPL_validation <- model_variables[validation_index,]
```

```
dim(JPL_data)
```

```
## [1] 958492     18
```

Model dataset has 851993 observations (rows) across 18 columns (including the predictor).

```
dim(JPL_validation)
```

```
## [1] 106501     18
```

Validation dataset has 213000 observations (rows) across 18 columns (including the predictor).

2.6 Variable Summary

2.6.1 id, class, neo_class

- *id* : The unique id for each object.
- *class* : classification of the objects accroding to their orbits.
- *neo_class* : The model Outcome Variable.

```
JPL_SmallBody_data <- data.frame(JPL_data)
summary(JPL_SmallBody_data[1:3])
```

```
##       id               class           neo_class
##  Length:958492      Length:958492      NEO: 21183
##  Class :character   Class :character   PHO:  1946
##  Mode  :character   Mode  :character   SO :935363
```

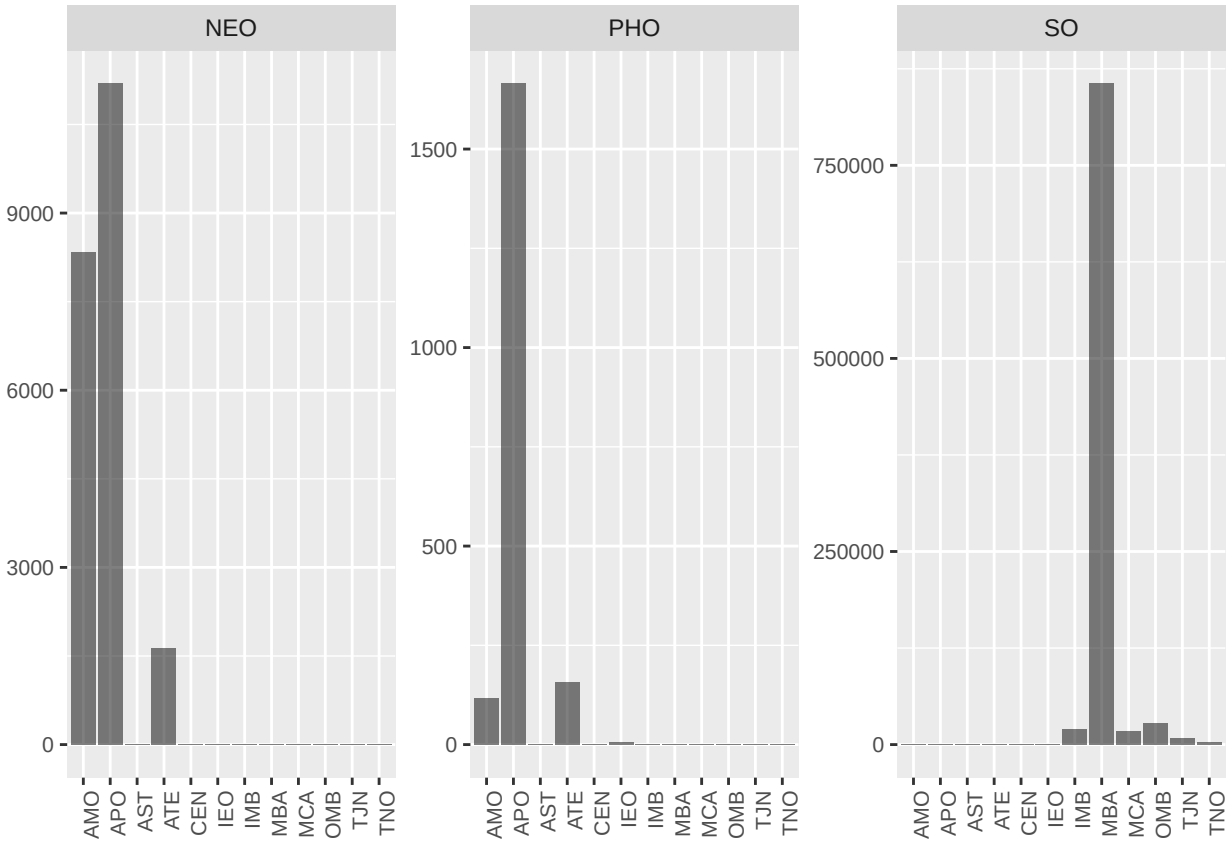
The table below shows the distribution of the Outcome Variable by each orbital class.

```
tab <- table(JPL_SmallBody_data[2:3])
tab %>% knitr::kable()
```

	NEO	PHO	SO
AMO	8337	118	0
APO	11193	1665	0
AST	0	0	62
ATE	1637	157	0
CEN	0	0	488
IEO	16	6	0
IMB	0	0	20477
MBA	0	0	855885
MCA	0	0	17986
OMB	0	0	28370
TJN	0	0	8895
TNO	0	0	3200

A visual representation of the differrent class by the neo_class.

```
tab %>% as.data.frame() %>%
  ggplot(aes(class,Freq)) +
  geom_bar(stat="identity", fill = 9, alpha=0.5) +
  facet_wrap(~neo_class, ncol=3,scales = "free") +
  theme(axis.text.x = element_text(angle=90,size=8, hjust=1),
        axis.text.y = element_text(size=8, lineheight = 1.5),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        legend.position = "bottom")
```



- The PHO and NEO have evidently a more similar orbit as compared to the SO. It might be possible to categorise our solar objects into these categories based on the observations that are available.

However, the "class" variable will not be used for further analysis.

2.6.2 H, e, t_jup, a

- H : Absolute Magnitude (M) is a measure of the luminosity of a celestial object, on an inverse logarithmic astronomical magnitude scale. In other words, the magnitude of an asteroid at zero phase angle and at unit heliocentric and geocentric. The more luminous an object, the smaller the numerical value of its absolute magnitude.
- e : Eccentricity e is the ratio of half the distance between the foci c to the semi-major axis $e = \frac{c}{2\alpha}$. An orbit with $e = 0$ is circular, $e = 1$ is parabolic and e between 0 and 1 is elliptic.
- t_jup : Tisserand Invariant is a value calculated from several orbital elements(semi-major axis, orbital eccentricity and inclination) of a relatively small object and a larger perturbing body, in this case Jupiter. This is used to distinguish asteroids (typically $T_J > 3$) from Jupiter-family comets (typically $2 < T_J < 3 $) amd the minor planet groups of damocloids (typically $T_J \leq 2$).
- a : Semi-Major Axis of an orbit ellipse is half the length of the major axis. For solor system bodies, this value is denoted by α

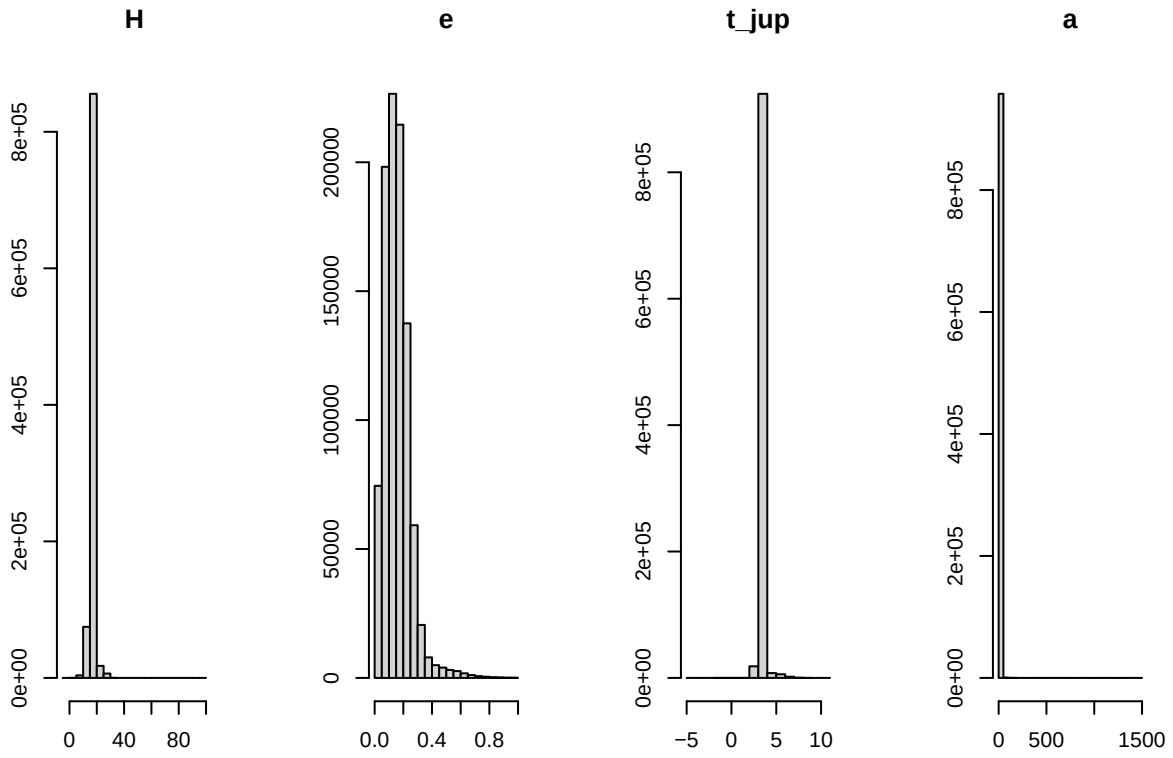
Summary

```
k <- 24576
summary(JPL_SmallBody_data[4:7])
```

```
##        H                 e              t_jup              a          
##  Min.   : -0.13   Min.   :0.0000   Min.   :-4.90   Min.   :   0.56  
##  1st Qu.: 16.20   1st Qu.:0.0924   1st Qu.: 3.21   1st Qu.:   2.39  
##  Median : 17.01   Median :0.1456   Median : 3.34   Median :   2.65  
##  Mean   : 17.04   Mean   :0.1565   Mean   : 3.38   Mean   :   2.87  
##  3rd Qu.: 17.90   3rd Qu.:0.2013   3rd Qu.: 3.50   3rd Qu.:   3.01  
##  Max.   : 99.99   Max.   :0.9969   Max.   :10.19   Max.   :1455.27
```

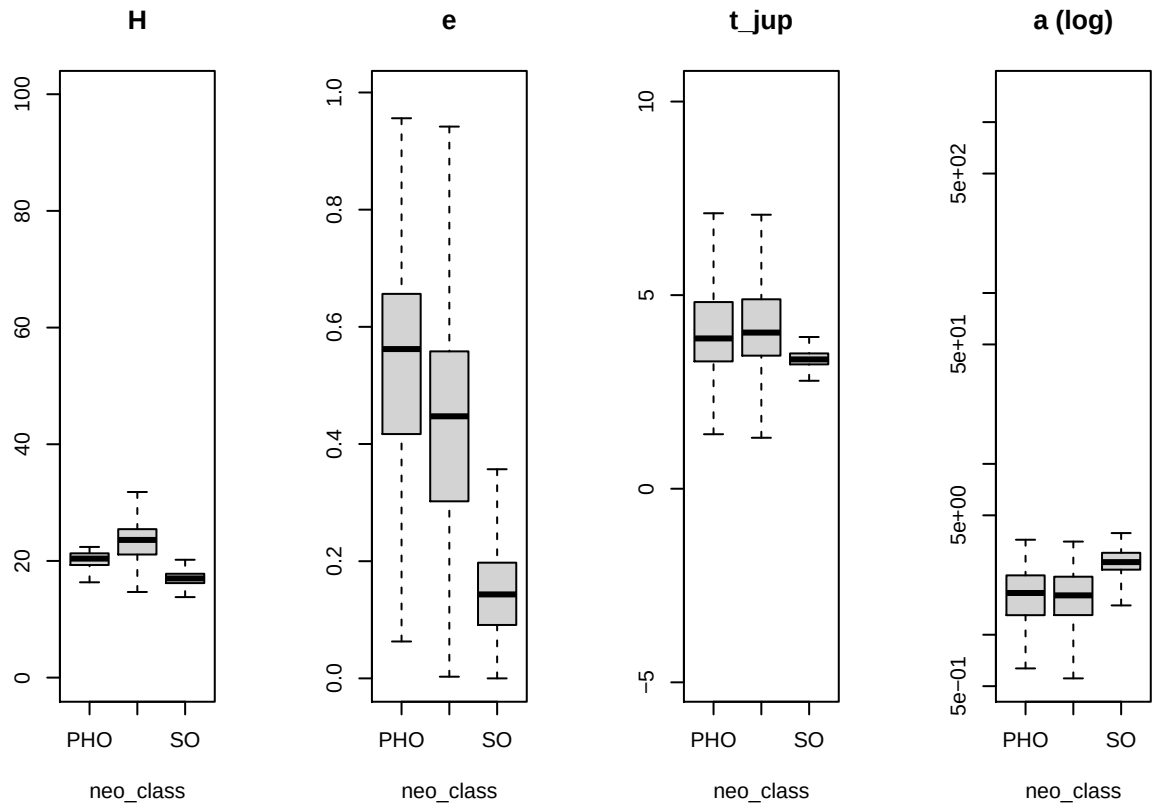
- The range of H is between -1.11 (more luminous) and ~100 (less luminous).
- The range of e, as expected, is between 0 and 1.
- The range of t_jup is between -4.90 (Damocloids) and 10.19 (Asteroids).

```
par(mfrow=c(1,4))
for(iter in 4:7) {
  hist(JPL_SmallBody_data[,iter], main=names(JPL_SmallBody_data)[iter],xlab='', ylab='')}
```



Variations of H, e, t_jup by the *neo_class* (Outcume Variable) :

```
par(mfrow=c(1,4))
X <- JPL_SmallBody_data[,3:7]
for(iter in 2:5) {dat <- list(PHO = X[X$neo_class == "PHO",iter],
                              NEO = X[X$neo_class == "NEO",iter],SO = X[X$neo_class == "SO",iter])
ifelse(iter != 5, boxplot(dat,xlab="neo_class", cex=0, main=str_c(names(X)[iter], "")),
boxplot(dat,xlab="neo_class", log="y", cex=0, main=str_c(names(X)[5]," (log)")))}
```

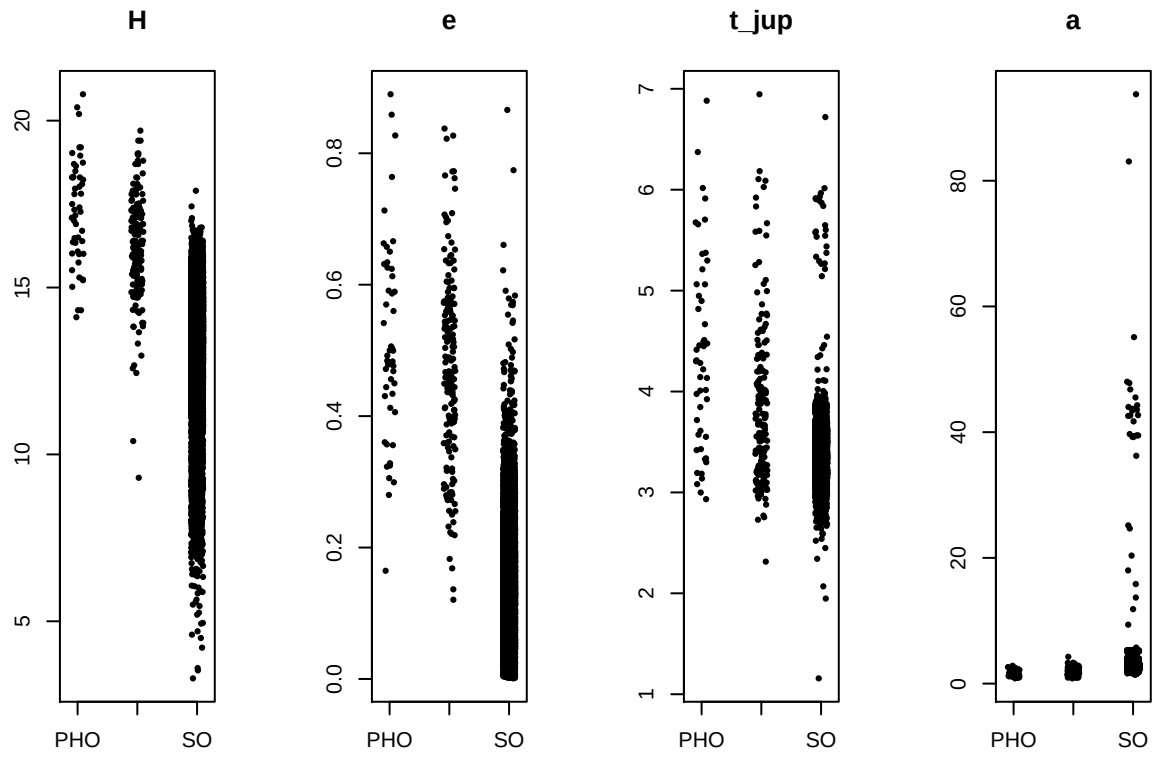


Observations:

- H , e and t_jup show some distinction (given a few outliers within) between the three prediction classes (PHO, NEO, SO).
- a shows distinction between the SO and the NEO,PHO groups in terms of range.

The stripchart below shows the spread of H, e, t_jup (of ~25,000 observations) in the neo_class :

```
par(mfrow=c(1,4))
X <- JPL_SmallBody_data[1:k,3:7]
for(iter in 2:5) { dat <- list(PHO = X[X$neo_class == "PHO",iter],
                               NEO = X[X$neo_class == "NEO",iter],SO = X[X$neo_class == "SO",iter])
stripchart(dat,vertical=TRUE,method="jitter", pch=16, col=1, main=str_c(names(X)[iter], ""))}
```



Observations: *(~25,000 observations)*

- H has a smaller range in *NEO,PHO* groups as compared to *SO*
- a has a lower range in *NEO,PHO* groups as compared to *SO*

2.6.3 q, ad, moid, moid_jup

- q : Perihelion Distance is the distance between the orbiting body and the sun at its closest apporach.
- ad : Aphelion Distance is the distance between the orbiting body and the sun when it is furthest away.
- $moid$: Earth Minimum Orbit Intersection Distance. It is the distance between the closest point of osculating orbits of two bodies, in this case with Earth and the other small body object.
- $moid\_jup$: Jupiter Minimum Orbit Intersection Distance. It is the distance between the closest point of osculating orbits of two bodies, in this case Jupiter and the other small body object.

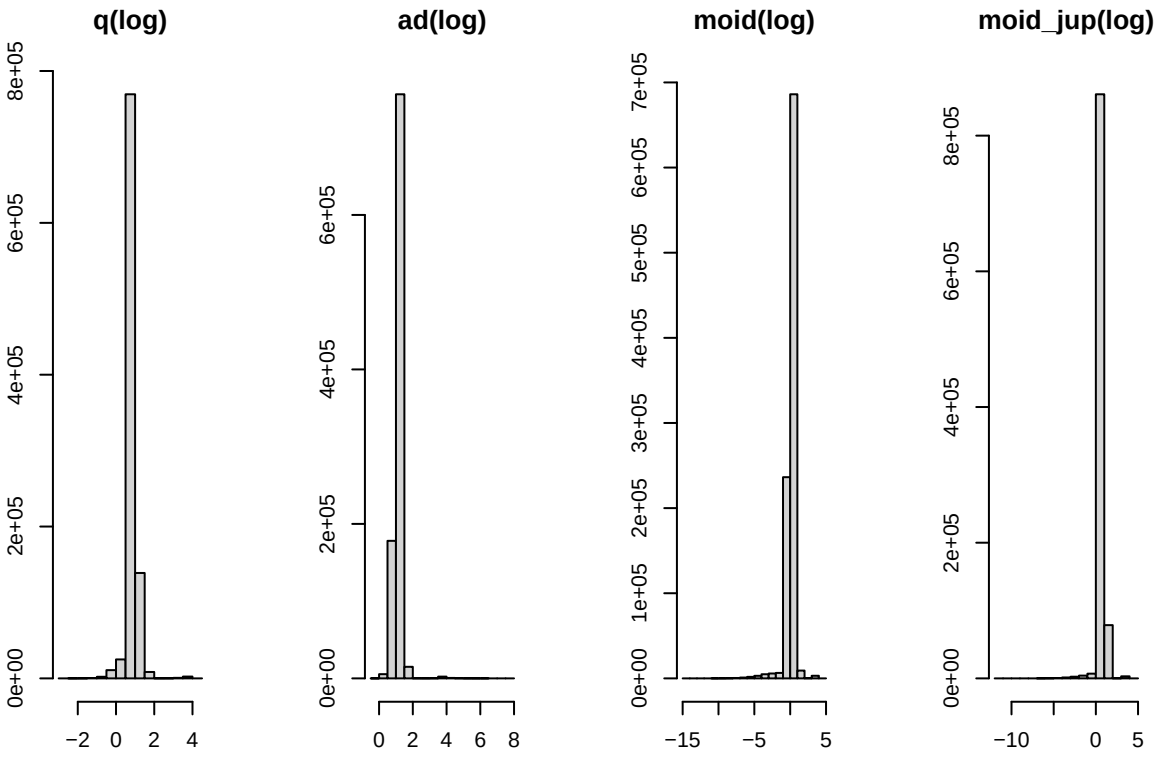
These variables are in au (astronomical unit) : $1\ au = 148,597,870.7\ km$

```
summary(JPL_SmallBody_data[8:11])
```

```
##        q                ad               moid          moid_jup
##  Min.   : 0.07   Min.   :   0.65   Min.   : 0.000   Min.   : 0.00
##  1st Qu.: 1.97   1st Qu.:   2.79   1st Qu.: 0.979   1st Qu.: 1.91
##  Median : 2.23   Median :   3.06   Median : 1.244   Median : 2.20
##  Mean   : 2.39   Mean   :   3.35   Mean   : 1.406   Mean   : 2.27
##  3rd Qu.: 2.58   3rd Qu.:   3.37   3rd Qu.: 1.595   3rd Qu.: 2.45
##  Max.   :80.40   Max.   :2906.08   Max.   :79.480   Max.   :75.78
```

- The range of $a, q, ad, moid, moid\_jup$ show the spread of the distance.

```
par(mfrow=c(1,4))
for(iter in 8:11) { hist(log(JPL_SmallBody_data[,iter]),
                         main=str_c(names(JPL_SmallBody_data)[iter],"(log)"),xlab='', ylab='')}
```



- The distribution chart shows that $a, q, ad, moid, moid\_jup$ have a very large number of values close to zero. *(PS: 0.01 au is still over a million kilometers)*

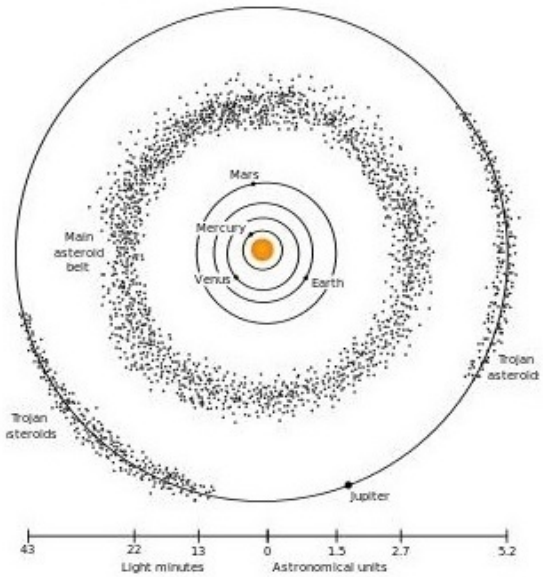
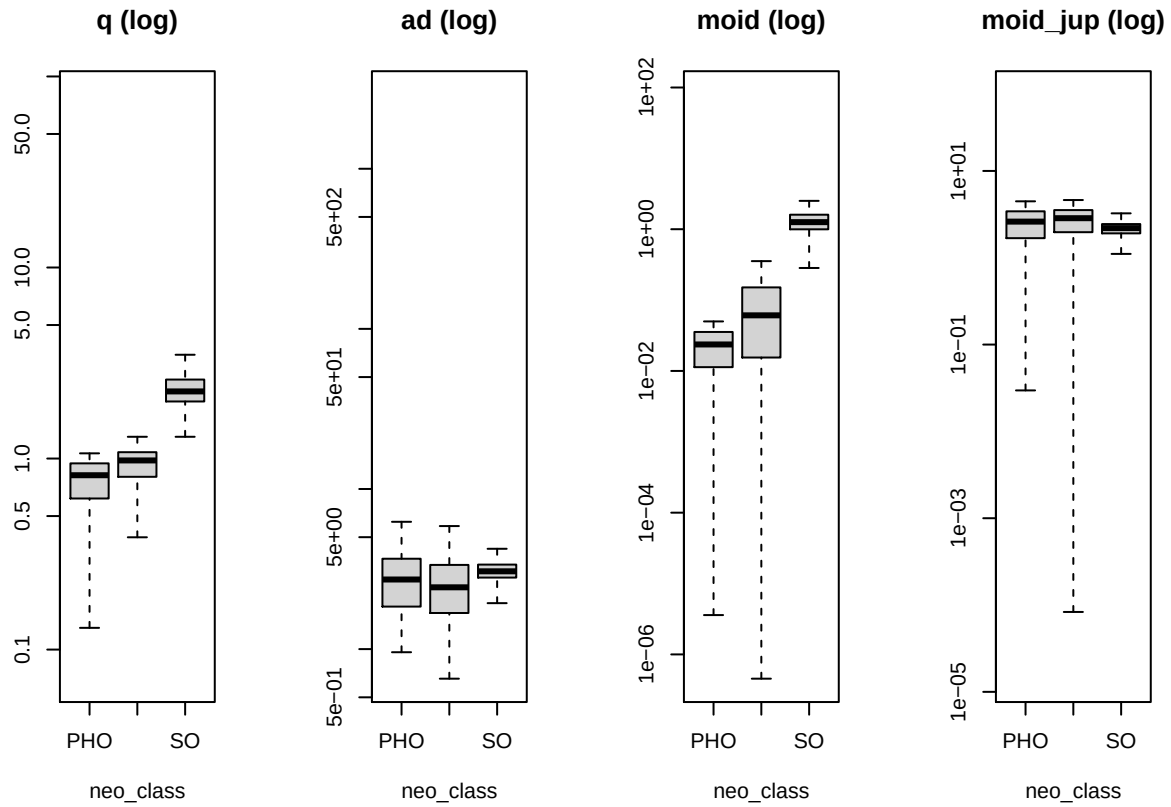


image source: wikipedia

```
par(mfrow=c(1,4))
X <- JPL_SmallBody_data %>% select(neo_class,q,ad,moid,moid_jup)
for(iter in 2:5) {  dat <- list(PHO = X[X$neo_class == "PHO",iter],
                                NEO = X[X$neo_class == "NEO",iter],SO = X[X$neo_class == "SO",iter])
boxplot(dat,xlab="neo_class",log='y',cex=0,  main=str_c(names(X)[iter], " (log)"))}
```

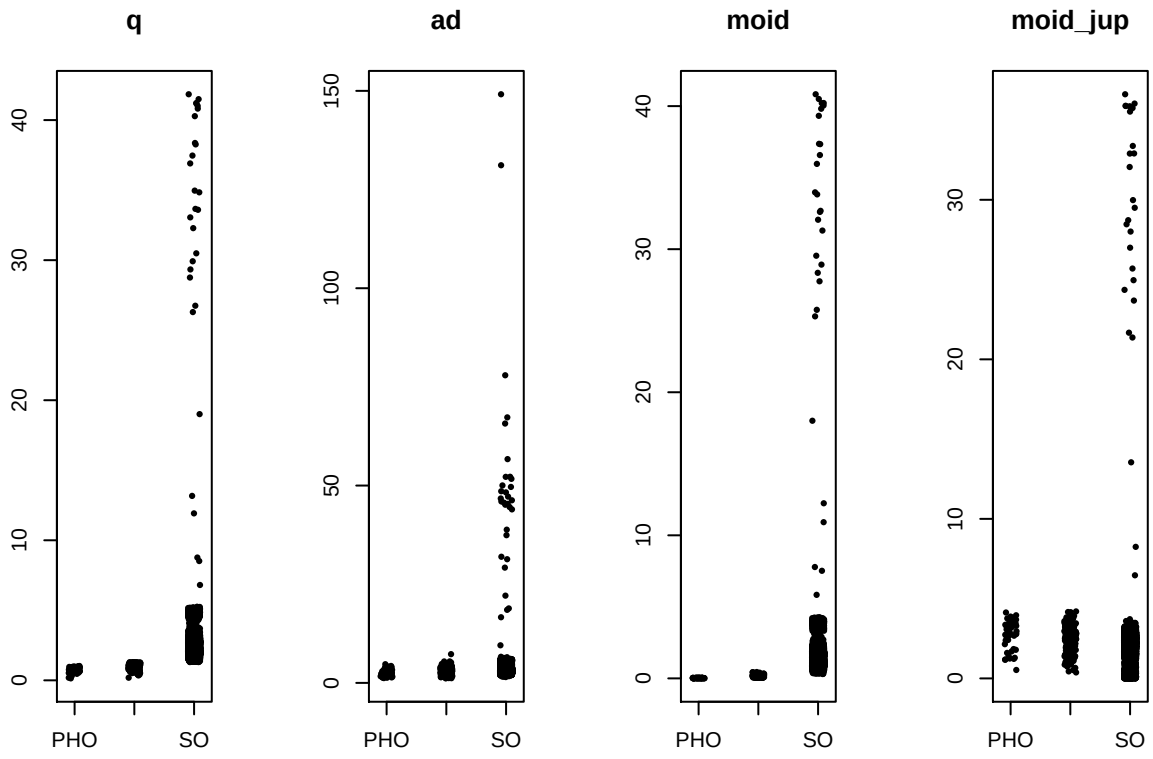


Observations:

- *q* does show distinction between the *SO* and the *NEO,PHO* groups in value terms, with those in *SO* having higher values.
- *ad* shows distinction between the *SO* and the *NEO,PHO* groups in terms of range.
- *moid* shows some distinction between *SO* and *NEO,PHO* in value terms, with those in *SO* having higher *moid*.

The stripchart below shows the spread of $a, q, ad, moid, moid\_jup$ (of ~25,000 observations) in the *neo_class* :

```
par(mfrow=c(1,4))
X <- JPL_SmallBody_data %>% select(neo_class,q,ad,moid,moid_jup)
X <- X[1:k,]
for(iter in 2:5) {  dat <- list(PHO = X[X$neo_class == "PHO",iter],
                                NEO = X[X$neo_class == "NEO",iter],SO = X[X$neo_class == "SO",iter])
stripchart(dat,vertical=TRUE,method="jitter", pch=16, col=1, main=str_c(names(X)[iter], ""))}
```



Observations: *(~25,000 observations)*

- $q, ad, moid$ have lower range in *NEO,PHO* groups as compared to *SO*

Derived Variables

A derived variables from $moid, moid\_jup$ is defined as:

- m_d is the difference of **Earth Minimum Orbit Intersection Distance** ($moid$) and **Jupiter Minimum Orbit Intersection Distance** ($moid\_jup$).

$$m\_d \;=\; moid\_jup \;-\; moid$$

The new variable will reflect in the Process Data chapter.

2.6.4 i, om, w

- i : Inclination is the angle between the vectors normal to the object orbit plane and the specified reference plane (typically elliptic and the equatorial plane).
- om : Longitude of the ascending node is the angle between the X-direction (typically the vernal equinox) and the point at which the body passes up (north) through the reference plane. Often denoted by Ω.
- w : Argument of perihelion is the angle (in the object orbit plane) between the ascending node line and perihelion measured in the direction of the object orbit. Often denoted by ω.

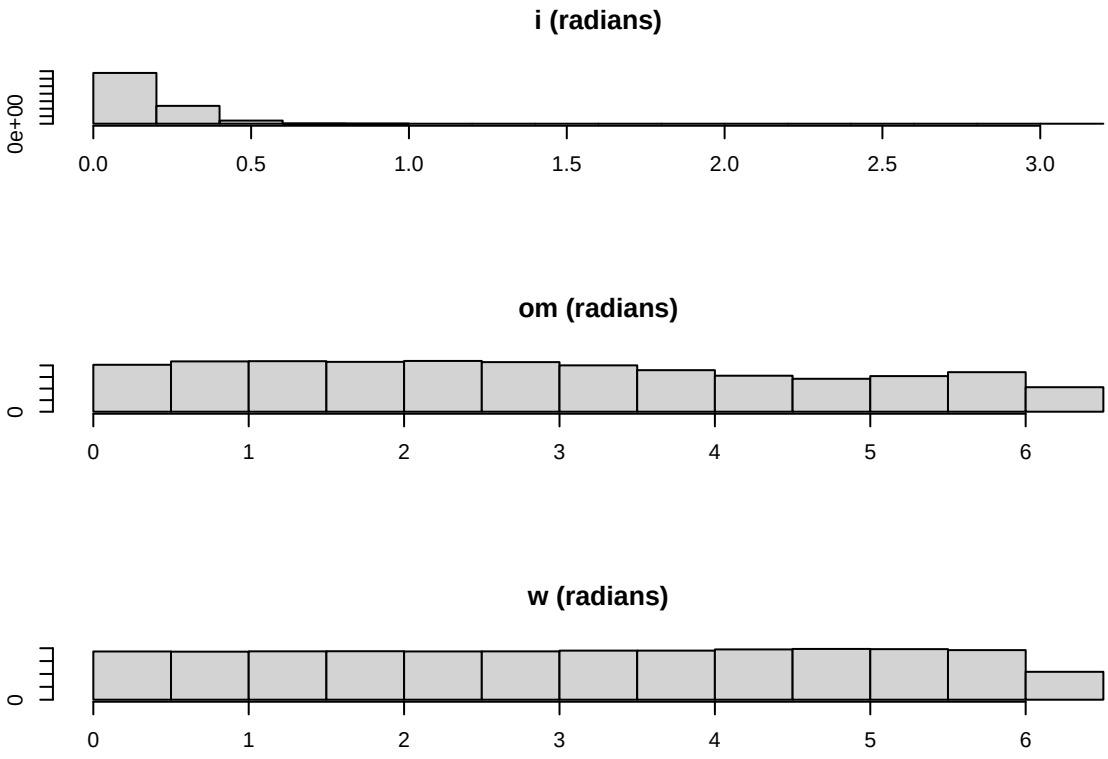
These variables are in degrees : $\pi\ rad = \ 180\ degrees$

```
summary(JPL_SmallBody_data[12:14]*pi/180)
```

```
##        i                  om              w
##  Min.   :0.00014   Min.   :0.00   Min.   :0.00
##  1st Qu.:0.07365   1st Qu.:1.41   1st Qu.:1.60
##  Median :0.13089   Median :2.79   Median :3.19
##  Mean   :0.15853   Mean   :2.94   Mean   :3.17
##  3rd Qu.:0.21697   3rd Qu.:4.41   3rd Qu.:4.74
##  Max.   :3.07137   Max.   :6.28   Max.   :6.28
```

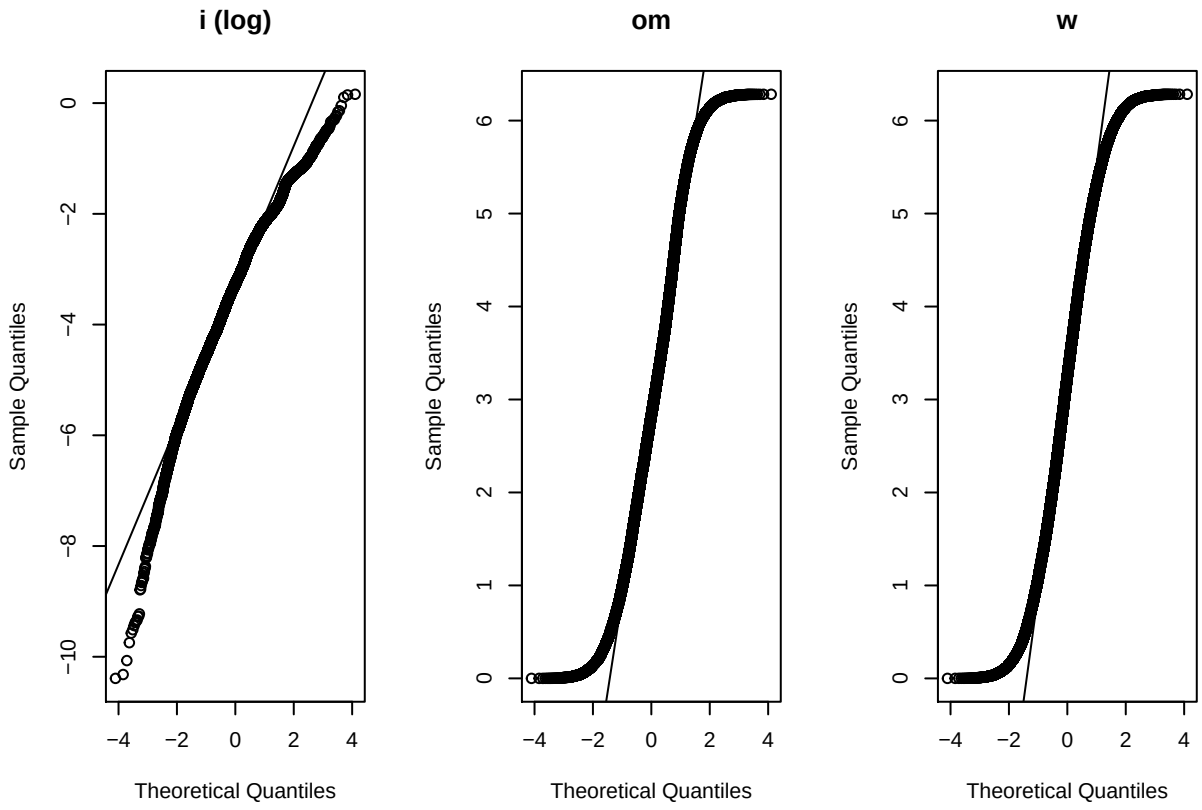
- i has a smaller range (lower maximum value) than om, w.

```
par(mfrow=c(3,1))
for(iter in 12:14) {
  hist((JPL_SmallBody_data[,iter]*pi/180),
       main=str_c(names(JPL_SmallBody_data)[iter]," (radians)"), xlab='', ylab='')}
```



- i is mostly distributed in the ~0.5 to 0 range.

```
par(mfrow=c(1,3))
qqnorm(log2(JPL_SmallBody_data[1:k,12]*pi/180),
       main=str_c(names(JPL_SmallBody_data)[12]," (log)"))
qqline(log2(JPL_SmallBody_data[1:k,12]*pi/180))
for(iter in 13:14) {
  qqnorm((JPL_SmallBody_data[1:k,iter]*pi/180), main=names(JPL_SmallBody_data)[iter])
  qqline((JPL_SmallBody_data[1:k,iter]*pi/180))}
```



Derived Variables

- **Inclination** (i), **Longitude of the ascending node** (om) and **Argument of perihelion** (w) is being converted to radians

$$\{i = i * \frac{pi}{180}\} \ , \{om = om * \frac{pi}{180}\} \ , \{w = w * \frac{pi}{180}\}$$

These changes will reflect in the Process Data chapter.

2.6.5 per, n, ma, rms

- *per* : Orbit period is the time required for an object to make a complete revolution along its orbit. A typical main belt asteroid has an orbit period of 4 years. Orbit period in earth days is used here.
- *n* : Mean Motion is the angular speed required for a body to make one orbit around an ideal ellipse with a specific semi-major axis. $\frac{(2*pi)}{Oribital\ period}$
- *ma* : Mean Anomaly is the product of an orbiting object mean motion and time past the perihelion passage.
- *rms* : Normalized RMS Of Orbit Fit.

```
summary(JPL_SmallBody_data[15:18])
```

```
##       per                 n                ma              rms        
##  Min.   :     151   Min.   :0.00002   Min.   :  0.0   Min.   :    0.0  
##  1st Qu.:    1351   1st Qu.:0.18884   1st Qu.: 81.4   1st Qu.:    0.5  
##  Median :    1578   Median :0.22820   Median :171.2   Median :    0.6  
##  Mean   :    2306   Mean   :0.23614   Mean   :175.3   Mean   :    0.6  
##  3rd Qu.:    1906   3rd Qu.:0.26656   3rd Qu.:269.0   3rd Qu.:    0.6  
##  Max.   :20277431   Max.   :2.38103   Max.   :369.1   Max.   :18129.0
```

- *rms* value rises after the 3rd quartile, (might be an outlier.)
- *per* value rises after the 3rd quartile, (might be an outlier.)

Outliers for *rms* :

```
JPL_SmallBody_data[JPL_SmallBody_data$rms > 100, 18]
```

```
## [1] 18129.00   402.83
```

Outliers for *per* :

```
table((JPL_SmallBody_data[JPL_SmallBody_data$per > (1000 *365), 3]))
```

```
##
## NEO PHO  SO
##   1   0 128
```

There is are *111* SmallBody Objects with an orbit period of *greater than 10 Centuries*, and one of them is also a NEO! (just for curiosity, which one?)

```
X <- (JPL_SmallBody_data[JPL_SmallBody_data$per > (1000 *365) &
                           JPL_SmallBody_data$neo_class == "NEO",1])
descriptive_data <- as.data.frame(read_csv('descriptive_data.csv'))
descriptive_data %>% filter(id==X) %>% knitr::kable()
```

id	full_name	producer	neo	pha	per_y	moid	first_obs	last_obs
bK17U52R	(2017 UR52)	Otto Matic	1	0	5787	0.31949	2017-10-29	2017-11-24

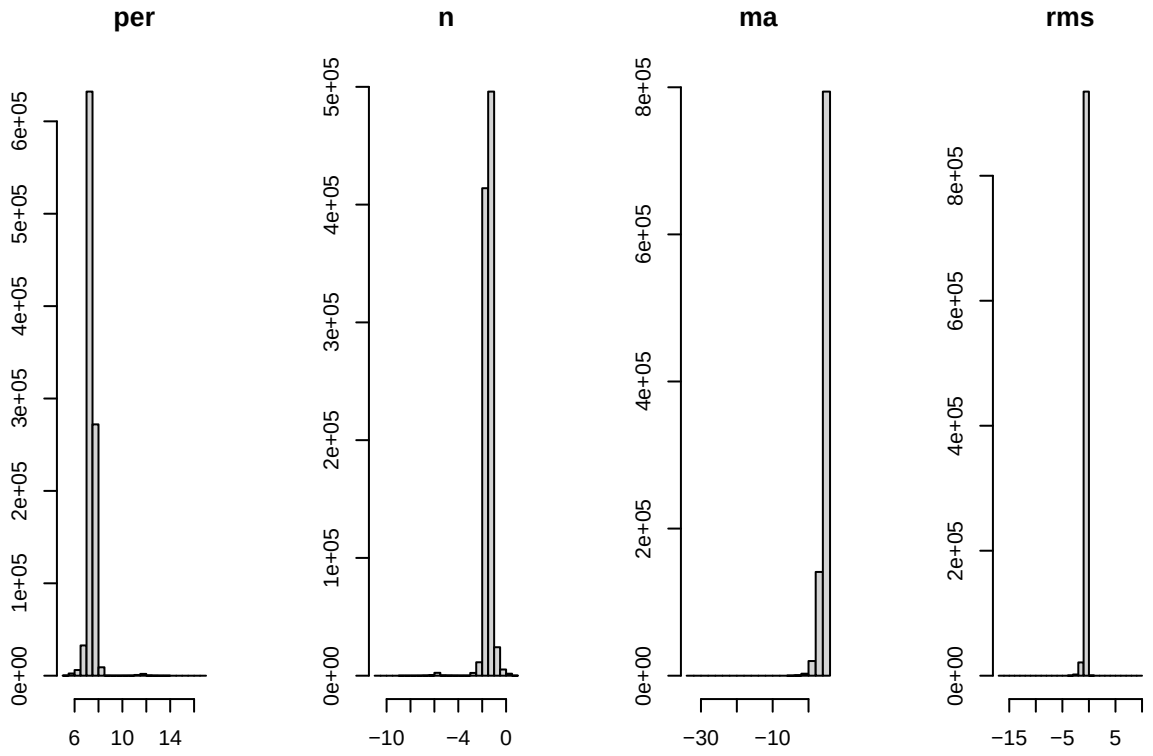
```
JPL_data[JPL_data$id == X, 1:11]  %>% knitr::kable()
```

id	class	neo_class	H	e	t_jup	a	q	ad	moid	moid_jup
bK17U52R	AMO	NEO	21.2	0.99614	-0.416	322.33	1.2426	643.41	0.31949	2.2985

Interestingly, *2017UR52* has an Earth Minimum Orbit Intersection Distance (*moid*) of 0.31949 *au* and has an Aphelion distance *ad* of 643.41 *au*.

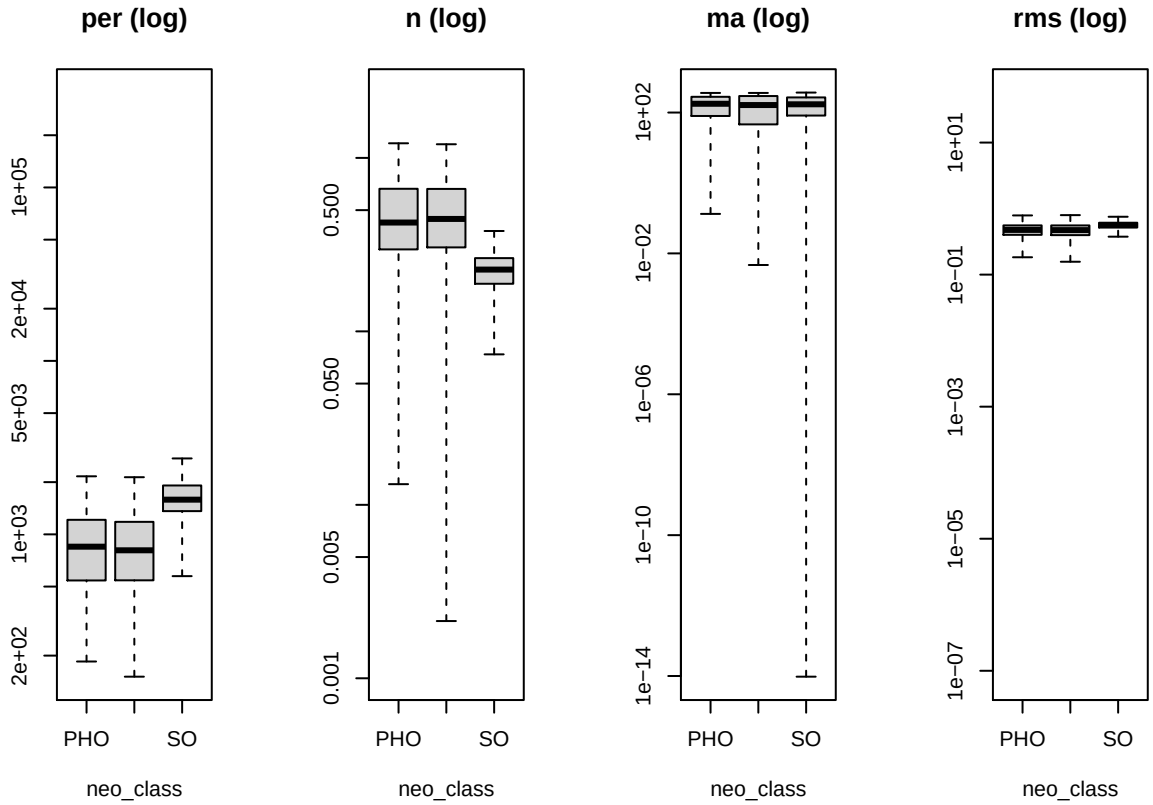
→ As a general rule, the outlier observations are not kept in the dataset, but since there are more variables the outliers of *per* > 10 *centuries* have been kept as other variables might be able to explain the outcome. Also, even a small body with a huge (in our time frames) orbital period might be a potential NEO (as in the case of *2017UR52*) or a PHO.

```
par(mfrow=c(1,4))
for(iter in 15:18) {
  hist(log(JPL_SmallBody_data[,iter]), main=names(JPL_SmallBody_data)[iter],xlab='', ylab='')}
```




```
par(mfrow=c(1,4))
X <- JPL_SmallBody_data %>% filter(rms <= 100, per <= 1000*365) %>%
```

```
  select(neo_class,per, n, ma, rms)
for(iter in 2:5) {
  dat <- list(PHO = X[X$neo_class == "PHO",iter],
              NEO = X[X$neo_class == "NEO",iter],SO = X[X$neo_class == "SO",iter])
boxplot(dat,xlab="neo_class",log='y',cex=0,  main=str_c(names(X)[iter]," (log)"))}
```

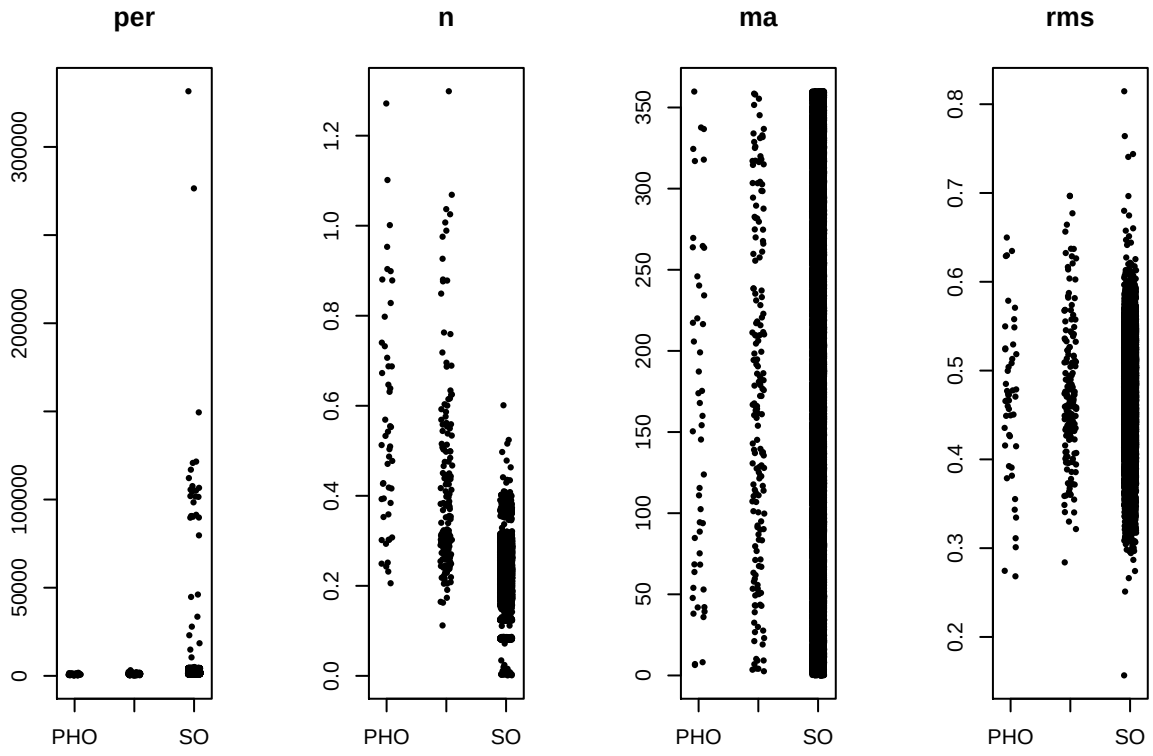


Observations:

- *per* for most *NEO,PHO* objects are smaller than *SO* and might imply that lower the orbit period (*per*) less the likelihood of the object being and *NEO,PHO*.

The stripchart below shows the spread of per, n, ma, rms (of ~25,000 observations) in the *neo_class* :

```
par(mfrow=c(1,4))
X <- JPL_SmallBody_data %>% select(neo_class,per, n, ma, rms)
X <- X[1:k,]
for(iter in 2:5) {
  dat <- list(PHO = X[X$neo_class == "PHO",iter],
              NEO = X[X$neo_class == "NEO",iter],SO = X[X$neo_class == "SO",iter])
  stripchart(dat,vertical=TRUE,method="jitter", pch=16, col=1, main=str_c(names(X)[iter], ""))}
```



Derived Variables

- **Time past the perihelion passage** $t_{perihelion}$ can be derived from **Mean Anamoly** (ma) and **Mean Motion**. (*reference to time is in days*)

$$t_{perihelion} = \frac{ma}{n}$$

- **Time past the Aphelion passage** ($t_{aphelion}$) is the difference between **Orbital period** (per) and **Time past the perihelion passage** (t_p). (*reference to time is in days*)

$$per = t_{aphelion} + t_{perihelion}$$

$$t_{perihelion} = per - t_{aphelion} \quad \text{or} \quad t_{aphelion} = per - \frac{ma}{n}$$

- **Ratio of perihelion period** (T_p) can thus be derived as

$$T_p = \frac{t_{perihelion}}{per}$$

$$=> T_p = 1 - \frac{t_{aphelion}}{per} \quad \text{or} \quad T_p = \frac{\frac{ma}{n}}{per}$$

These new variables will reflect in the Process Data chapter.

2.7 Variable Means

Comparison of variable means by *neo_class* :

```
pp <- JPL_SmallBody_data[,3:18]
m_comp <- data_frame()
for(iter in 2:15){
   Mean_PHO <- round(mean(pp[pp$neo_class=="PHO",iter]),2)
   Mean_NEO <- round(mean(pp[pp$neo_class=="NEO",iter]),2)
   Mean_SO <- round(mean(pp[pp$neo_class=="SO",iter]),2)
   m_comp <- bind_rows(m_comp,
                       data_frame(Variable=colnames(pp[iter]),
                                  PHO = Mean_PHO,NEO = Mean_NEO,SO = Mean_SO))}
m_comp  %>% knitr::kable()
```

Variable	PHO	NEO	SO
H	20.11	23.30	16.89
e	0.54	0.43	0.15
t_jup	4.18	4.27	3.36
a	1.79	1.77	2.90
q	0.76	0.93	2.43
ad	2.81	2.62	3.37
moid	0.02	0.10	1.44
moid_jup	2.54	2.71	2.26
i	13.98	12.28	9.00
om	174.47	172.90	168.44
w	180.27	182.38	181.56
per	914.54	992.27	2338.50
n	0.52	0.52	0.23
ma	179.22	171.02	175.38

Observations:

- *per* group means in descending order - *SO,NEO,PHO.*
- *ma* group means in descending order - *PHO,SO,NEO.*
- *H* group means in descending order - *NEO,PHO,SO.*
- *q* group means in descending order - *SO,NEO,PHO.*
- *i* group means in descending order - *PHO,NEO,SO.*

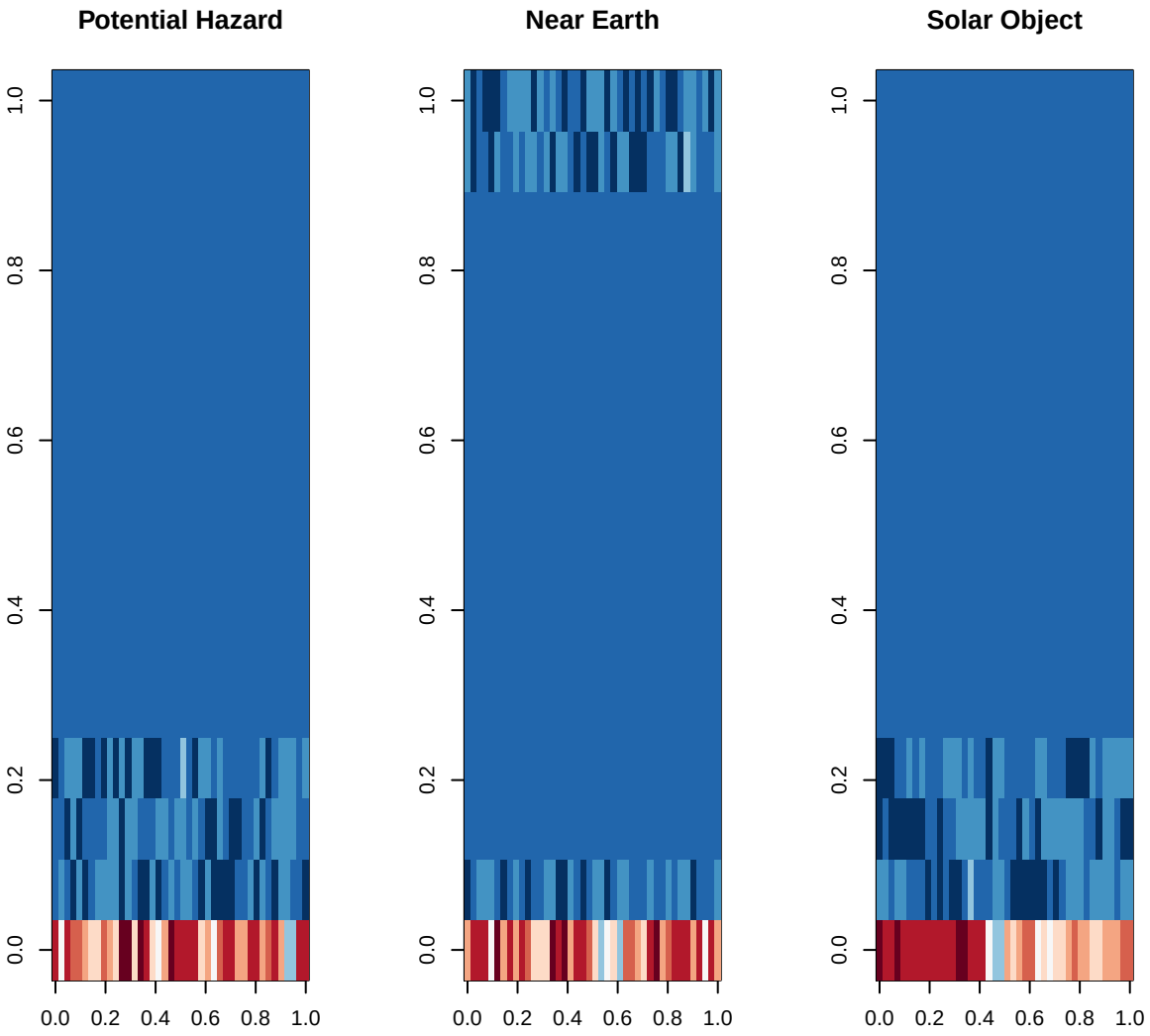
These observations show that the variables display a pattern in their values.

2.8 Distance

But, a picture is worth a thousand words.

The JPL_SmallBody_data dataset is normalized by columns, for each *neo_class*, and then ordered according to the results of the clustering algorithm for a visual cue.

```
x <- JPL_SmallBody_data[,3:18]

#extract and sweep different categories
ph <- x[x$neo_class =="PHO",-1] %>% as.matrix() %>% sweep(.,2, colMeans(.))
ne <- x[x$neo_class =="NEO",-1] %>% as.matrix() %>% sweep(.,2, colMeans(.))
so <- x[x$neo_class =="SO",-1] %>% as.matrix() %>% sweep(.,2, colMeans(.))

#sweep the data matrix
x <-  as.matrix(x[,-1])
x <- sweep(x,2, colMeans(x))

k <- 42
#hclusters based on data and the transpose
ph_1 <- hclust(dist(ph[1:k,]))
ph_2 <- hclust(dist(t(ph[1:k,])))
ne_1 <- hclust(dist(ne[1:k,]))
ne_2 <- hclust(dist(t(ne[1:k,])))
so_1 <- hclust(dist(so[1:k,]))
so_2 <- hclust(dist(t(so[1:k,])))

#image of the hclust
par(mfrow=c(1,3))
image(x[ph_1$order, ph_2$order],
  col=RColorBrewer::brewer.pal(11,'RdBu'), main = "Potential Hazard")

image(x[ne_1$order, ne_2$order],
  col=RColorBrewer::brewer.pal(11,'RdBu'), main = "Near Earth")

image(x[so_1$order, so_2$order],
  col=RColorBrewer::brewer.pal(11,'RdBu'), main = "Solar Object")
```



Observations:

- The image gives a clear visual cue on how the variables in the data provide a clear distinction between the individual categories of the Outcome variable *neo_class*.

3 Train & Test sets

```
set.seed(131825431, sample.kind = "Rounding")
# test set will be 10% of training set.
test_index <- createDataPartition(JPL_data$neo_class, times = 1, p = 0.1, list = FALSE)
JPL_train <- JPL_data[-test_index,]
JPL_test <- JPL_data[-test_index,]
```

3.1 Process Data

The variables are modified as mentioned above for the datasets to be used for training and testing. This data processing will also be done for the validation set.

```
data_process <- function(preData, tChoice){
  #convert to radians and add additional derived variables
    preData <- preData %>%
    mutate(om=om*(pi/180), w=w*(pi/180), i=as.numeric(i*(pi/180)),
           t_to_p = as.numeric((per - (ma/n))), a=log2(a),
           m_d = as.numeric(moid_jup-moid))
    preData <- preData %>% mutate(T_p = as.numeric(round(1-(t_to_p/per),4)))
    preData <- preData %>% select(neo_class,H,q,moid,m_d,e,t_jup,a,i,T_p)
    preData$neo_class <- as.factor(preData$neo_class)

    #return as matrix format
    if(tChoice == 1){
      preData <- data.frame(preData)
      x_matrix <- as.matrix(preData[,-1])
      y_matrix <- factor(preData[,1])
      processData <- list(X = x_matrix, Y = y_matrix)}
    #return as data table
    if(tChoice == 2) processData <- preData
    return (processData)
}
```

3.2 Inaccuracy Details function

```
descriptive_data <- data.table::fread("descriptive_data.csv", na.strings = c(NA,"NA",""))
descriptive_data <- as.data.frame(descriptive_data)

#function to collate inaccurately classified observations.
inaccurate_details <- tibble()
inaccuracy_check <- function(incorrect_p,method,dat){
  idetails <- data.frame()
  for(each in incorrect_p){
    a <- (dat[each,1])
    idetails <- rbind(idetails,data.frame(model_name = method,
                                          descriptive_data[descriptive_data$id==str_trim(a),]))}
  return(idetails)}
```

4 Model Evaluations

4.1 Evaluation metrics

4.1.1 Defination

The confusion matrix

	Actual Positive	**Actual Negative**
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

Recall, 1-FPR, and Precision

			Definition	Probability representation
sensitivity	TPR	Recall	$\frac{\text{TP}}{\text{TP+FN}}$	$\Pr(\hat{Y} = 1 \mid Y = 1)$
specificity	TNR	1-FPR	$\frac{\text{TN}}{\text{TN+FP}}$	$\Pr(\hat{Y} = 0 \mid Y = 0)$
specificity	PPV	Precision	$\frac{\text{TP}}{\text{TP+FP}}$	$\Pr(Y = 1 \mid \hat{Y} = 1)$

F1 Score

This is the harmonic average of Precision and Recall and is given as :

$$2 \times \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

4.1.2 Evaluation Function

In this case study the confusionmatrix function of the caret package is used. The following function takes the confusion matrix and adds the Recall and the F1 scores to a tibble for the results.

```
#function to collate the results
results_process<- function(x,name){
  x1 <- data_frame(model_name=name,Object="NEO",
                   Recall=round(x[1,6],3),F1=round(x[1,7],3))
  x2 <- data_frame(model_name=name,Object="PHO",
                   Recall=round(x[2,6],3),F1=round(x[2,7],3))
  x3 <- data_frame(model_name=name,Object="SO",
                   Recall=round(x[3,6],3),F1=round(x[3,7],3))
  return(bind_rows(x1,x2,x3))
}
```

4.2 PCA

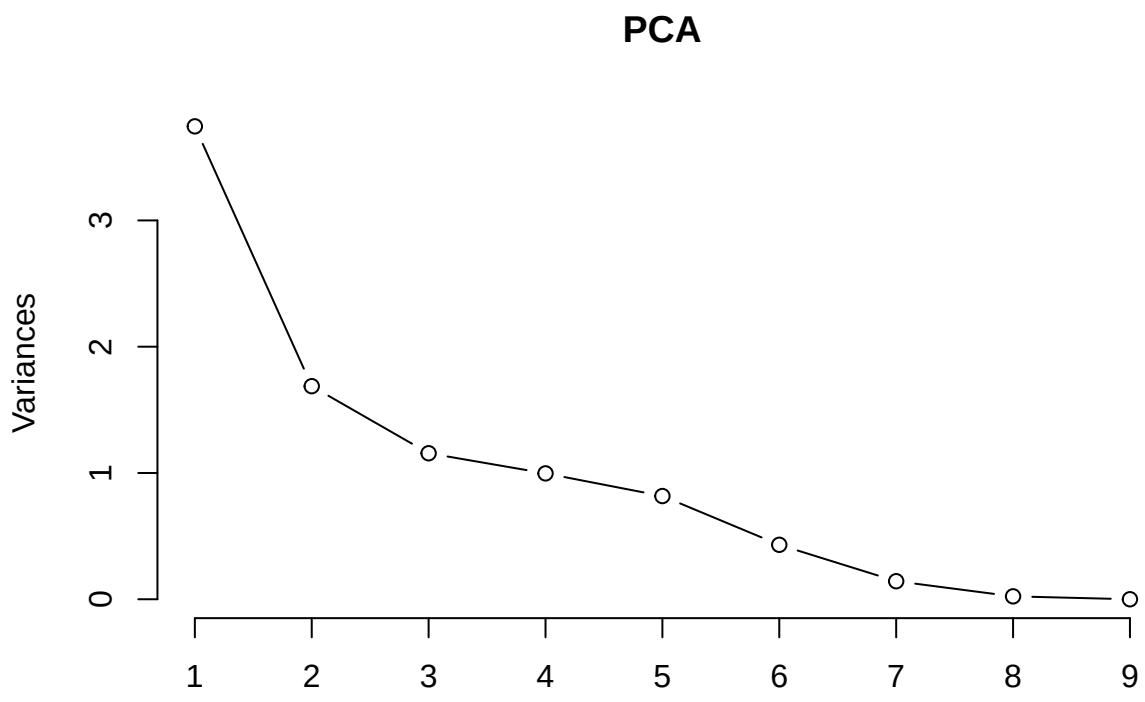
The principal component Analysis assists in reducing the complexity of model being fit by transforming the data, with the distance between the rows preserved but with the variance of the variables in decreasing order.

```
mat_train <- data_process(JPL_train,1)
mat_test <- data_process(JPL_test,1)
set.seed(131825431, sample.kind = "Rounding")
pca <- prcomp(mat_train$X, center = TRUE, scale. = TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1   PC2   PC3   PC4    PC5    PC6    PC7     PC8
## Standard deviation     1.935 1.299 1.075 0.998 0.9040 0.6568 0.3781 0.15187
## Proportion of Variance 0.416 0.187 0.128 0.111 0.0908 0.0479 0.0159 0.00256
## Cumulative Proportion  0.416 0.604 0.732 0.843 0.9336 0.9815 0.9974 0.99998
##                            PC9
## Standard deviation     0.01258
## Proportion of Variance 0.00002
## Cumulative Proportion  1.00000
```

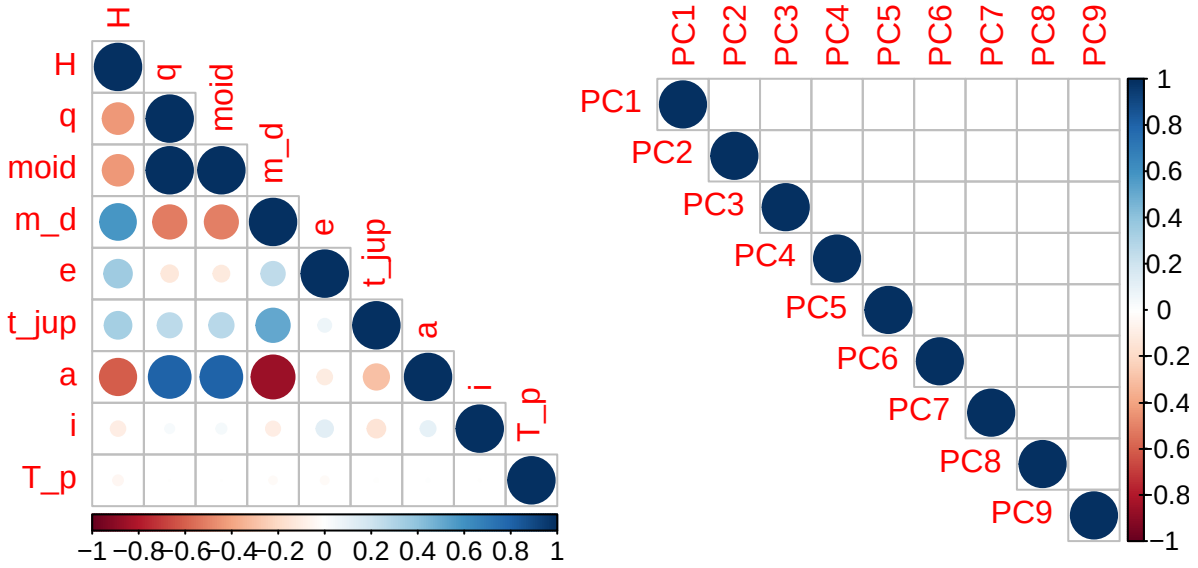
- The first six principal components account for about 80% of the variance in our training set.
- The plot below shows the corresponding variances for the PCAs.

```
par(mfrow=c(1,1))
screeplot(pca,type = "line", main="PCA")
```



Correlation Plot shows the correlation between the variables.

```
par(mfrow=c(1,2))
corrplot(cor(mat_train$X), method="circle", type="lower")
corrplot(cor(pca$x), method="circle", type="upper")
```



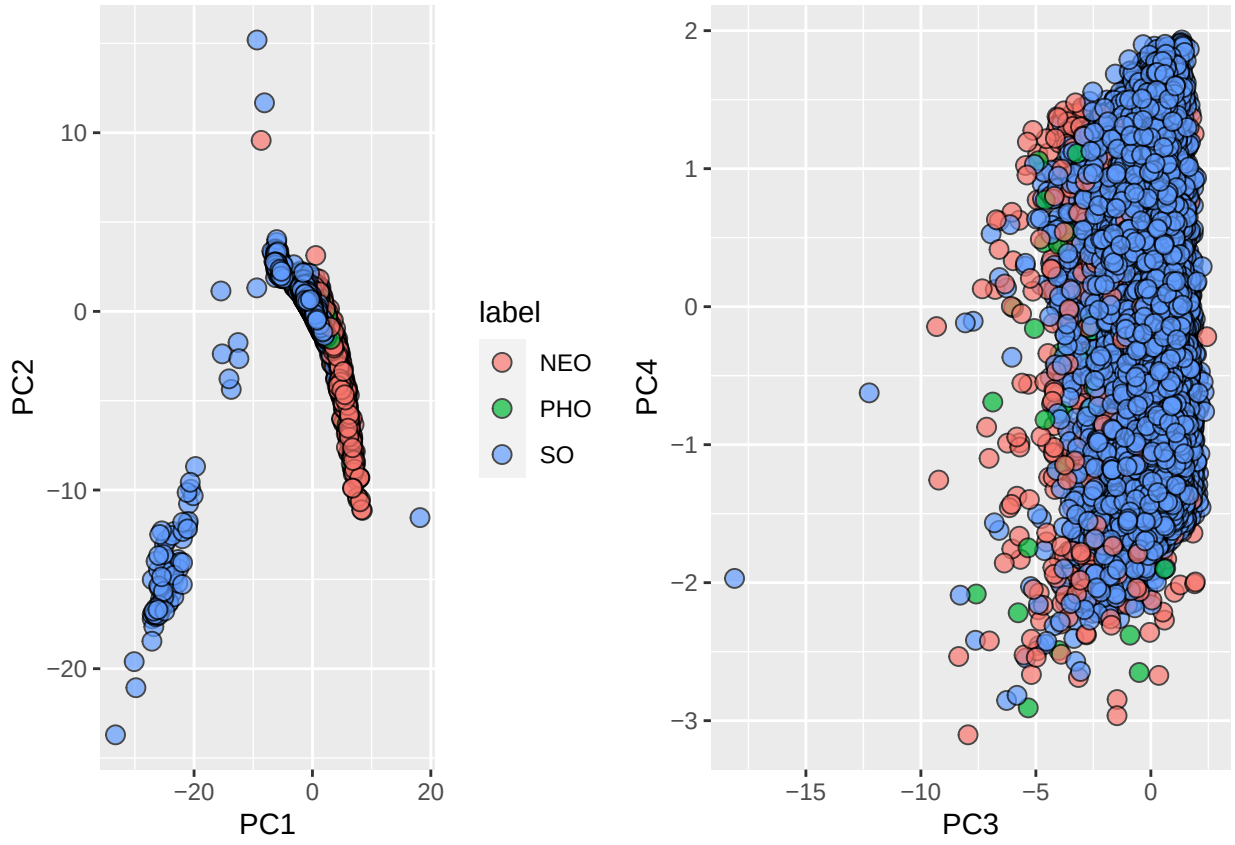
Observations:

- The qa_d variable, which is the differnce between the Perihelion Distance q and the Aphelion Distance ad has a correlation of 0.92 with the Semi-Major Axis a.
- There is no corelation in the pca components.

Plot of the the first 4 components of the PCA together even on a random sample, gives a visual cue on the information on the *neo_class*.

Note: random sample of 24,576 observations used.

```
set.seed(131825431, sample.kind = "Rounding")
par(mfrow=c(1,1))
p1 <- data.frame(PC1 = pca$x[,1], PC2 = pca$x[,2], label= mat_train$Y) %>%
  sample_n(24576) %>%  ggplot(aes(PC1,PC2, fill = label)) +
  geom_point(cex=3,pch=21,alpha=0.7) + theme(legend.position = "right")
p2 <- data.frame(PC3 = pca$x[,3], PC4 = pca$x[,4],label= mat_train$Y) %>%
  sample_n(24576) %>% ggplot(aes(PC3,PC4, fill = label)) +
  geom_point(cex=3,pch=21,alpha=0.7) + theme(legend.position = "hide")
gridExtra::grid.arrange(p1,p2, ncol=2, nrow=1)
```



4.2.1 Multinom

The first model that is tried is a multinom model using the nnet library. It fits multinomial log-linear models via neural networks.

- The first 7 components of the PCA are used, which account for 98.6% of the variance.
- The test set is rotated using the rotation matrix of PCA.

Model Fitting :

```
#first 7 components
k <- 7
train_pca <- data.frame(pca$x[,1:k], neo_class= as.factor(mat_train$Y))
test_pca <- as.matrix(sweep(mat_test$X,2, colMeans(mat_test$X))) %*% pca$rotation
test_pca <- data.frame(test_pca[,1:k], neo_class= as.factor(mat_test$Y))
set.seed(131825431, sample.kind = "Rounding")
fit_pca_multinom <- multinom(neo_class ~ ., data = train_pca, sumn=1)
```

Model Prediction and the confusion matrix:

```
y_hat_pca_multinom <- predict(fit_pca_multinom,test_pca, type = "class")
cm <- confusionMatrix(y_hat_pca_multinom, test_pca$neo_class)
results <- results_process(as.data.frame(cm$byClass),"PCA Multinom")
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO    989     59      0
##        PHO    102     61      0
##        SO   17973   1631 841826
```

```
cat("Accuracy : ", cm$overall["Accuracy"], "\n")
```

```
## Accuracy :  0.97709
```

- The multinom model was able to identify some of the categories, however since the 7 PCA components used explain about 99.7% of the variablily. The accuracy that we have is almost 98%, but it is the Sensitivity in individual categories (SO has a Recall of 1) that matters as well in this case.

```
results %>% filter (model_name == "PCA Multinom")  %>% knitr::kable()
```

model_name	Object	Recall	F1
PCA Multinom	NEO	0.052	0.098
PCA Multinom	PHO	0.035	0.064
PCA Multinom	SO	1.000	0.988

The PCA components, though explain the variablitly with lesser variables (components) will not be used as the Recall of 0.99 is the objective, and though accuracy is high the requiremnt of Recall is not being met.

4.3 Naive Bayes

The second model is the Naive Bayes model, a general genrative model.

- In a binary case, the smallest true error ahchieved is determined by the Bayes rule, which is a decision rule that is based on true conditional probabilty, and is given by $p(X) = Pr(Y = 1|X = x)$.
- Predictors are assumed to be independent within each class label.
- Numeric (metric) predictors are handled by assuming that they follow Gaussian distribution, given the class label, and are modelled with Poisson distribution.
- Prediction is done by the type class.

Model Fitting, Prediction and the confusion matrix:

```
set.seed(131825431, sample.kind = "Rounding")
fit_nb <- naive_bayes(mat_train$X,mat_train$Y, usekernel=FALSE, usepoisson = TRUE)
y_hat_nb <- predict(fit_nb, mat_test$X, type = "class")
cm <- confusionMatrix(y_hat_nb, mat_test$Y)
results <-  bind_rows(results,results_process(as.data.frame(cm$byClass),"Naive Bayes"))
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO  18774    540   4390
##        PHO    246   1211    172
##        SO      44      0 837264
```

- This model does a bit with a higher F1 and Recall and is able to classify NEO and PHO better.

```
results %>% filter (model_name == "Naive Bayes")  %>% knitr::kable()
```

model_name	Object	Recall	F1
Naive Bayes	NEO	0.985	0.878
Naive Bayes	PHO	0.692	0.717
Naive Bayes	SO	0.995	0.997

- The Recall for all three categories is above 0.9 but the F1 Score (the harmonic average of Precision and Recall) for PHO is below 0.5

4.4 Ensemble

In order to get the right direction on the model which will fit well with the data set, an Ensemble of 6 models is run on a smaller sample.

- LDA : Linear discriminant Analysis. The assumption here is correlation structure is the same for all classes, which thus reduces the number of parameters that are required to estimate.
- QDA : Quadratic discriminant Analysis. This is a version of Naive Bayes in which the assumption is that the distributions $p_{X|Y=1}(x)$ and $p_{X|Y=0}(x)$ are multivariate normal.
- KNN : k-Nearest Neighbor Classification. The k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote.
- rpart : Recursive Partitioning and Regression Trees. These are decision trees that work by predicting an outcome variable Y by partitioning the predictors.
- svmRadialCost : Support Vector Machine with radial cost. In simple terms, the SVM consturcts a linear model with the largest possible margin given to the data points.
- wsrf : Forest of Weighted Subspace Decision Trees. C4.5-based trees (Quinlan (1993)) are grown by wsrf, where binary split is used for continuous predictors (variables) and k-way split for categorical ones.

Note: *sample of 16,384 observations for training and 1638 obeservations for testing is used.*

Model Fitting :

```
set.seed(131825431, sample.kind = "Rounding")
ensemble_train <- data_process(JPL_train[1:16348,],1)
ensemble_test <- data_process(JPL_test[1:1638,],1)
results_ensemble <- data.frame()

models <- c("lda", "qda", "knn", "rpart", "svmRadialCost", "wsrf")

fits <- lapply(models, function(model){
  cat(":- ",model, "\n")
  if (model %in% c("lda","qda")){
    train(ensemble_train$X,ensemble_train$Y, method=model, prior = c(1,1,1)/3)
  }
  else train(ensemble_train$X,ensemble_train$Y, method=model,
             trControl = trainControl(method = "cv", number=5))
})
```

Prediction and the confusion matrix :

```
names(fits) <- models
pred <- sapply(fits,function(object){  predict(object, ensemble_test$X) })

for (each in 1:ncol(pred)){
  cm <- confusionMatrix(factor(pred[,each]), ensemble_test$Y)
  results_ensemble <- bind_rows(results_ensemble,
                                results_process(as.data.frame(cm$byClass),colnames(pred)[each]))
}
```

Ensemble Results:

```
results_ensemble %>% filter (model_name == "lda")  %>% knitr::kable()
```

model_name	Object	Recall	F1
lda	NEO	0.400	0.480
lda	PHO	0.667	0.500
lda	SO	0.998	0.997

```
results_ensemble %>% filter (model_name == "qda")  %>% knitr::kable()
```

model_name	Object	Recall	F1
qda	NEO	1.000	0.732
qda	PHO	1.000	1.000
qda	SO	0.993	0.997

```
results_ensemble %>% filter (model_name == "knn")  %>% knitr::kable()
```

model_name	Object	Recall	F1
knn	NEO	0.600	0.643
knn	PHO	0.000	NA
knn	SO	0.999	0.998

```
results_ensemble %>% filter (model_name == "rpart")  %>% knitr::kable()
```

model_name	Object	Recall	F1
rpart	NEO	1	1
rpart	PHO	1	1
rpart	SO	1	1

```
results_ensemble %>% filter (model_name == "svmRadialCost")  %>% knitr::kable()
```

model_name	Object	Recall	F1
svmRadialCost	NEO	1.000	0.938
svmRadialCost	PHO	0.333	0.500
svmRadialCost	SO	1.000	1.000

```
results_ensemble %>% filter (model_name == "wsrf")  %>% knitr::kable()
```

model_name	Object	Recall	F1
wsrf	NEO	1	1
wsrf	PHO	1	1
wsrf	SO	1	1

- The qda model has a Recall of 1 for the PHO category.
- The rpart and the wsrf models fit well with this subset of the data.

Thus, from the ensemble it seems that the decision tree based models will be the ones which achieve the objective of 0.99 Recall and 0.99 F1.

4.5 Decision Trees

4.5.1 Rpart

Recursive Partitioning and Regression Trees.

These are decision trees that predict an outcome variable Y by partitioning the predictors.

The idea is to build a decision tree and, at the end of each node, obtain a predictor $\hat{y}$. The prediction space is partitioned into J non-overlapping regions, $R_1, R_2, ..., R_J$ and then for any predictor x that falls within the region R_j, estimate $f(x)$ with the average of the training observations y_i for which the associated predictor x_i is also in R_j. Regression trees create partitions recursively. Description of how the partition is picked for further partitions and when to stop is given below:

$$R_1(j,s) = \{x|x_j < s\} \text{ and } R_2(j,s) = \{x|x_j \geq s\}$$

- x is the partition to be split.
- j is the predictor.
- s is the value that defines the 2 partitions.

j and s are picked up by finding the pair that minimizes the residual sum of squares (RSS):

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

This is then applied recursively to the new regions R_1 and R_2. After partitioning the predictor space into regions, in each region a prediction is made using the observations in that region.

Model Parameters :

- Complexity Parameter : The residual sum of squares must improve by a factor of cp for the new partition to be added. Large values of cp will therefore force the algorithm to stop earlier which results in fewer nodes.

Complexity Parameter :

```
set.seed(131825431, sample.kind = "Rounding")
train_rpart <- train(mat_train$X,mat_train$Y, method="rpart",
                     tuneGrid = data.frame(cp=seq(0.0,0.05, len=25)),
                     trControl = trainControl(method = "cv", number=5))
rpart_cp <- train_rpart$bestTune
rpart_cp
```

```
##          cp
## 20 0.039583
```

Variable Importance :

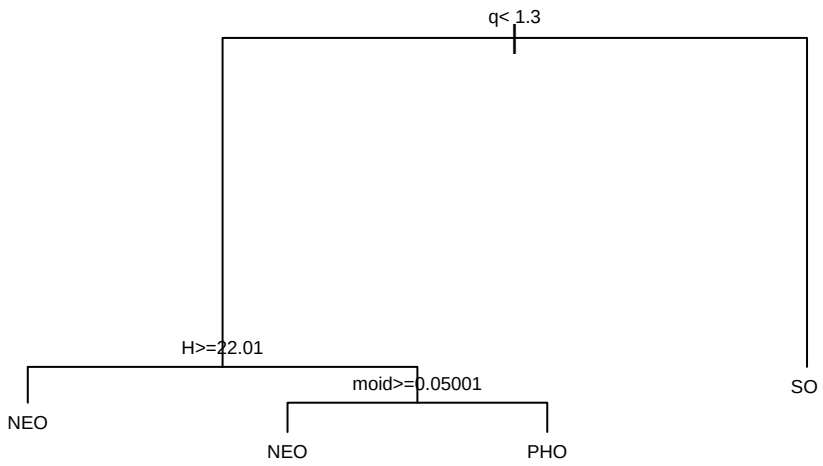
```
train_rpart$finalModel$variable.importance
```

```
##     moid        q        H        a      m_d        e        i    t_jup
## 38705.79 37569.11 26177.31 19022.00 18035.70 15214.81   190.64    59.35
```

- $moid, q, H, a, m\_d, e, i, t\_jup$ are used in the rpart fit.

Visual cue on the decision tree:

```
plot(train_rpart$finalModel, margin=0.1)
text(train_rpart$finalModel, cex=0.6)
```



Model Fitting, Prediction and the confusion matrix:

```
rtrain <- data_process(JPL_train,2)
rtest <- data_process(JPL_test,2)
set.seed(131825431, sample.kind = "Rounding")
fit_rpart <- rpart(neo_class ~. , data=rtrain, parms = list(split = "gini"),
             control = rpart.control(cp = rpart_cp))
predcit_rpart <- predict(fit_rpart, rtest)
labels <- colnames(predcit_rpart)[apply(predcit_rpart, 1, which.max)]
cm <- confusionMatrix(factor(labels), rtest$neo_class)
results <- bind_rows(results,results_process(as.data.frame(cm$byClass),"rpart"))
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO  19026     15      0
##        PHO     38   1736      0
##        SO       0      0 841826
```

Results :

```
results %>% filter (model_name == "rpart")  %>% knitr::kable()
```

model_name	Object	Recall	F1
rpart	NEO	0.998	0.999
rpart	PHO	0.991	0.985
rpart	SO	1.000	1.000

- The Recall is ≥ 0.99, but the F1 score for PHO is 0.985

However, the rpart is known to not be very flexible and is unstable to changes in the training data. Random Forest is used next to overcome some of these shortcomings.

```
inaccurate_details <- rbind(inaccurate_details, inaccuracy_check(which(
  labels != rtest$neo_class ),"rpart",JPL_test))
```

4.5.2 Random Forest

Random forests are a popular machine learning approach that addresses the shortcomings of decision trees and improving prediction performance and reduce instability by *averaging* multiple decision trees (a forest of trees constructed with randomness).

This is done by

- Bagging : Generate many predictors, each using regression or classification trees.
- Bootstrap : To induce randomness of individual trees bootstrap is used.

The individual trees **randomly** different, and the combination of trees is the **forest**.

randomForest implements Breiman random forest algorithm (based on Breiman and Cutler original Fortran code) for classification and regression.

Tuning Function :

```
gettuned <- function(method_name,grid,dat){
  set.seed(131825431, sample.kind = "Rounding")
  control <- trainControl(method='cv', number = 5)
  train(x = dat$X[2048:8192,], y = dat$Y[2048:8192], nTree=50,
        method=method_name, tuneGrid = grid, trControl = control)$bestTune}
```

Model Parameters :

- mtry : Number of variables randomly sampled as candidates at each split. The function *gettuned* above gets the mtry for the bestTuned randomForest of a small sample size of 6,144 observations.
- strata : A (factor) variable that is used for stratified sampling.
- sampsize : Size of sample to draw.
- nodesize : Minimum size of terminal nodes. For classification it is 1.
- nTree : Number of trees to grow.

```
rf_mtry <- gettuned("rf",data.frame(mtry=c(2,3,4,5,6)),mat_train)
```

Model Fitting :

```
set.seed(131825431, sample.kind = "Rounding")
rf_train <- data_process(JPL_train,2)
rf_test <- data_process(JPL_test,2)
fit_rf <- randomForest(neo_class ~ .,data=rf_train, mtry=rf_mtry$mtry,
                       nodesize = 1, strata = neo_class, nTree=rf_mtry$mtry*50,
                       sampsize = 100000, replace=FALSE)
```

Gini Index is defined as

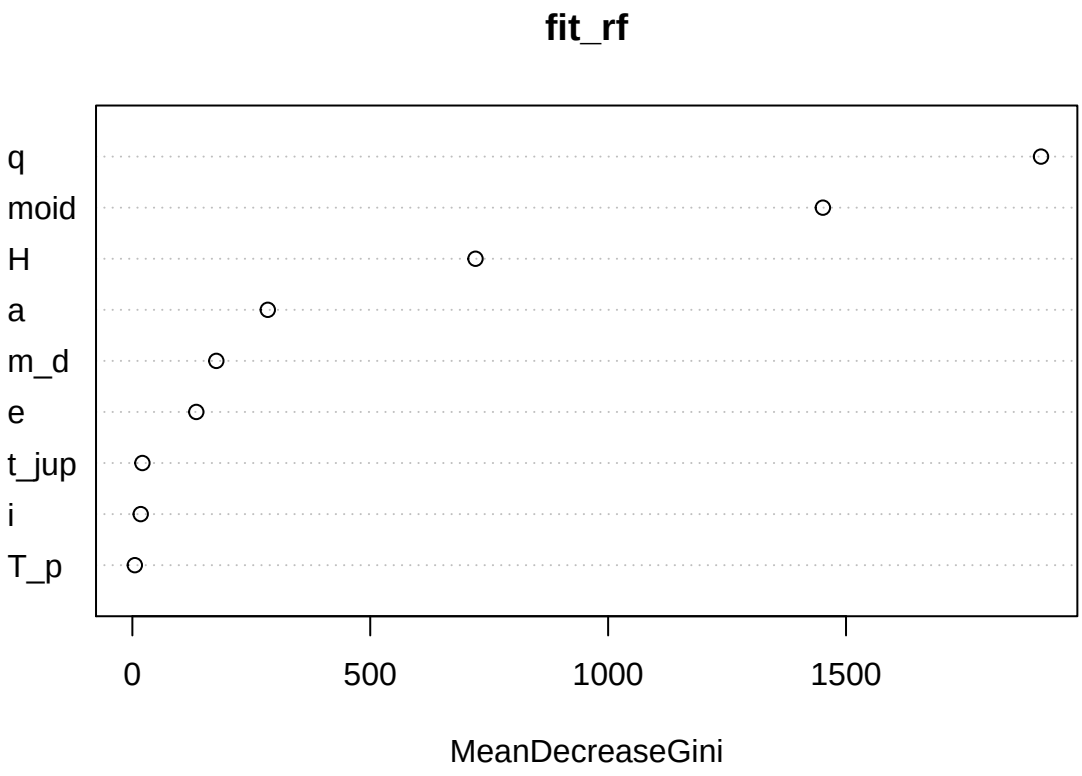
$$\text{Gini}(j) = \sum_{k=1}^{K} \hat{p}_{j,k}(1 - \hat{p}_{j,k})$$

- $\hat{p}_{j,k}$ as the proportion of observations in partition
- j that are of class k.

GINI importance measures the gain of purity by partitions of a given variable and assists in the decision function of the random forest to select from available partitions.

Variable Importance of the last measure:

```
varImpPlot(fit_rf)
```



- $q, moid, H$ achieved the highest meanDecrease, in the last measure.

Prediction and the confusion matrix:

```
y_hat_rf <- predict(fit_rf,rf_test)

cm <- confusionMatrix(y_hat_rf, rf_test$neo_class)
results <-  bind_rows(results,results_process(as.data.frame(cm$byClass),"Random Forest"))
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO  19035     19      1
##        PHO     29   1732      0
##        SO       0      0 841825
```

Results :

```
results %>% filter (model_name == "Random Forest")  %>% knitr::kable()
```

model_name	Object	Recall	F1
Random Forest	NEO	0.998	0.999
Random Forest	PHO	0.989	0.986
Random Forest	SO	1.000	1.000

- The Recall is ≥ 0.99, but the F1 scores for PHO is 0.986
- In the rpart NEO had 38 misses and PHO had 15 misses. In the Random Forest though the NEO misses have been lesser the PHO misses have gone up.

This might be caused due to the structure of the data which has over 97.6% of SO. The shortcoming of not being able to associate weights to categories of importance is approached with the WSRF model.

```
incorrect_p <- which(y_hat_rf != rf_test$neo_class )
inaccurate_details <- rbind(inaccurate_details,inaccuracy_check(
  incorrect_p,'Random Forest',JPL_test))
```

4.5.3 WSRF

Forest of Weighted Subspace Decision Trees

Model Parameters :

- mtry : number of trial predictors for a split (mtry).
- nTree : the number of trees to train.
- minNode : minimum number of distinct row references to split a node.
- weights : TRUE for weighted subspace selection.
- importance : Importance of predictors.

```
wsrf_mtry <- gettuned("wsrf",data.frame(mtry=c(2,3,4,5,6)),mat_train)
```

Model Fitting:

```
fit_wsrf <- wsrf(x = mat_train$X, y = mat_train$Y,
                 ntree = wsrf_mtry$mtry*100, mtry=wsrf_mtry$mtry+1,
                 weights=TRUE,  importance = TRUE, parallel = TRUE)
```

Variable Importance:

```
fit_wsrf$importance
```

```
##                 1           2        3 MeanDecreaseAccuracy MeanDecreaseIGR
## H      4.3853e-01 -0.00601504 0.000000           9.6773e-03         1.60424
## q      9.7405e-01  0.95160149 0.024147           4.7017e-02         2.15759
## moid   2.0908e-03  0.96285192 0.000000           1.9989e-03         2.34837
## m_d    1.5502e-03  0.00228858 0.000000           3.8873e-05         1.02320
## e      2.2230e-03  0.00486232 0.000000           5.9164e-05         1.03253
## t_jup  9.9007e-04  0.00059395 0.000000           2.3076e-05         0.88126
## a      2.1911e-04  0.00070127 0.000000           6.2228e-06         1.04245
## i      5.4128e-05  0.00053115 0.000000           2.2687e-06         1.75072
## T_p   -7.2673e-06 -0.00060043 0.000000          -1.3859e-06         1.68448
```

Prediction and confudsion matrix :

```
y_hat_wsrf <- predict(fit_wsrf, mat_test$X)
y_hat_wsrf_m <- as.matrix(y_hat_wsrf$class)
y_hat_wsrf_m <- ifelse(y_hat_wsrf_m == 1, 'NEO',ifelse(y_hat_wsrf_m == 2, 'PHO','SO'))

cm <- confusionMatrix(as.factor(y_hat_wsrf_m),mat_test$Y)
results <- bind_rows(results,results_process(as.data.frame(cm$byClass),"WSRF"))
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO  19059      8      0
##        PHO      5   1743      0
##        SO       0      0 841826
```

Inaccurate Predictions :

```
inaccurate_details <- rbind(inaccurate_details, inaccuracy_check(which(
  mat_test$Y != y_hat_wsrf_m),"WSRF",JPL_test))
inaccurate_details %>% filter (model_name == "WSRF") %>% .[,3:10] %>% tibble() %>% knitr::kable()
```

full_name	producer	neo	pha	per_y	moid	first_obs	last_obs
(2000 UQ30)	Otto Matic	1	1	2.30864	0.01241	2000-10-31	2000-12-22
(2000 WC1)	Otto Matic	1	1	0.82478	0.00383	2000-11-16	2014-11-16
(2001 QJ96)	Otto Matic	1	1	2.00941	0.00073	2001-08-23	2015-08-17
(2001 XP31)	Otto Matic	1	1	1.22747	0.01139	2001-12-11	2018-02-04
(2003 HN16)	Otto Matic	1	1	1.46226	0.02121	2003-04-27	2015-11-06
(2004 HG12)	Otto Matic	1	1	1.73195	0.02671	2004-04-21	2004-05-04
(2012 KN11)	Otto Matic	1	0	3.03543	0.02116	2012-05-19	2012-06-13
(2014 WR367)	Otto Matic	1	1	3.77863	0.01492	2014-11-21	2014-12-15
(2015 RE36)	Otto Matic	1	0	1.49899	0.00052	2012-09-19	2018-10-03
(2017 WX12)	Otto Matic	1	1	3.20251	0.02155	2017-11-20	2018-03-18
(2018 PH22)	Otto Matic	1	0	2.12302	0.04166	2018-07-13	2021-03-24
(2019 KD7)	Otto Matic	1	0	0.91890	0.03411	2019-05-26	2019-06-02
(2020 WD5)	Otto Matic	1	0	2.45773	0.04348	2020-11-18	2020-12-01

Results :

```
results %>% filter (model_name == "WSRF")  %>% knitr::kable()
```

model_name	Object	Recall	F1
WSRF	NEO	1.000	1.000
WSRF	PHO	0.995	0.996
WSRF	SO	1.000	1.000

- The Recall and F1 scores are all > 0.99
- Using a model which implements weights does increase the sensitivity on the outcome prediction.
- However there still are a total of 8 misses in PHO and 5 misses in NEO.

This might be partially explained by the incapability to attach weights manually, only a logical value is accepted in this model.

Thus the shortcoming of not being able to associate weights manually to each category according to their importance is approached with the Ranger model.

4.5.4 Ranger

Ranger is a fast implementation of random forests (Breiman 2001) or recursive partitioning, particularly suited for high dimensional data. Classification and regression forests are implemented as in the original Random Forest (Breiman 2001). Includes implementations of extremely randomized trees (Geurts et al. 2006) and quantile regression forests (Meinshausen 2006).

Model Parameters :

- classification is set to TRUE and accordingly min.node.size is set to 1.
- importance : Variable importance mode.
- splitrule : For classification the default "gini".
- regularization.usedepth : The depth in regularization.
- num.threads : Number of threads / number of CPUs.
- class.weights : Weights for the outcome classes (in order of the factor levels) in the splitting rule (cost sensitive learning). For classification the weights are also applied in the majority vote in terminal nodes.

Model Fitting :

```
set.seed(131825431, sample.kind = "Rounding")
fit_ranger <- ranger(x = mat_train$X, y = mat_train$Y,
                     classification = TRUE,
                     min.node.size = 1,
                     mtry=rf_mtry$mtry+1,
                     num.trees = rf_mtry$mtry*50,
                     splitrule = 'gini',
                     importance = 'impurity',
                     regularization.usedepth = TRUE,
                     # additional weight of ~ 40:60:0  ~ 2:3:0 ratio from prime numbers.
                     class.weights = c(7,11,0),
                     num.threads = 6, verbose = TRUE)
```

```
## Growing trees.. Progress: 90%. Estimated remaining time: 3 seconds.
```

Variable Importance :

```
fit_ranger$variable.importance
```

```
##        H        q     moid      m_d        e    t_jup        a        i
## 28286.37 67546.53 44144.09  2240.86  1995.23   172.15  4308.85   836.59
##      T_p
##   157.08
```

Prediction and the confusion matrix :

```
y_hat_ranger <- predict(fit_ranger, mat_test$X)

cm <- confusionMatrix(y_hat_ranger$predictions, mat_test$Y)
results <- bind_rows(results,results_process(as.data.frame(cm$byClass),"Ranger"))
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO  19064      0      0
##        PHO      0   1751      0
##        SO       0      0 841826
```

- **Unity is achieved.** No misses.

```
cm$byClass
```

```
##            Sensitivity Specificity Pos Pred Value Neg Pred Value Precision
## Class: NEO           1           1              1              1         1
## Class: PHO           1           1              1              1         1
## Class: SO            1           1              1              1         1
##            Recall F1 Prevalence Detection Rate Detection Prevalence
## Class: NEO      1  1  0.0220996      0.0220996            0.0220996
## Class: PHO      1  1  0.0020298      0.0020298            0.0020298
## Class: SO       1  1  0.9758706      0.9758706            0.9758706
##            Balanced Accuracy
## Class: NEO                 1
## Class: PHO                 1
## Class: SO                  1
```

- **100% accuracy on the training set**
- Prevalance shows the ditribution of the *neo_class* factors as determined on the orginal data (a 97.6% for SO, 0.002 for PHO and 0.022 for NEO)

Results :

```
results %>% filter (model_name == "Ranger")  %>% knitr::kable()
```

model_name	Object	Recall	F1
Ranger	NEO	1	1
Ranger	PHO	1	1
Ranger	SO	1	1

- This will be one of the models that will be used for the Final validation set.

4.6 Gradient Boosting

Generalized Boosted Regression Modeling (GBM)

Boosting is the process of iteratively adding basis functions in a greedy fashion so that each additional basis function further reduces the selected loss function. This implementation closely follows Friedman Gradient Boosting Machine (Friedman, 2001).

The **GBM** implements the generalized boosted modeling framework.

Model Parameters :

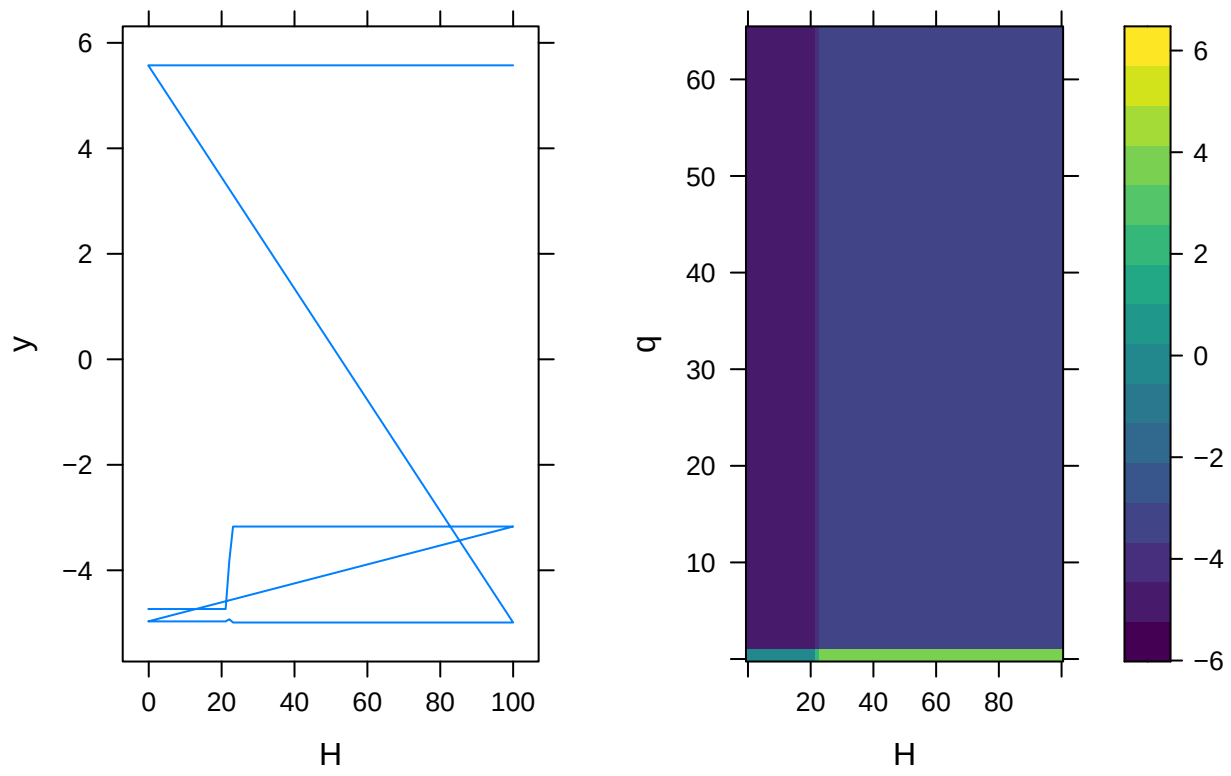
- n.trees : Integer specifying the total number of trees to fit.
- class.stratify.cv : Logical indicating whether or not the cross-validation should be stratified by class. The purpose of stratifying the cross-validation is to help avoiding situations in which training sets do not contain all classes.
- bag.fraction : The fraction of the training set observations randomly selected to propose the next tree in the expansion. This introduces randomnesses into the model fit.
- train.fraction : The first train.fraction * nrows(data) observations are used to fit the gbm and the remainder are used for computing out-of-sample estimates of the loss function.
- shrinkage : A shrinkage parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction.
- keep.data : a logical variable indicating whether to keep the data and an index of the data stored with the object.
- interaction.depth : Integer specifying the maximum depth of each tree (i.e., the highest level of variable interactions allowed). A value of 1 implies an additive model. A value of 3 implies a model with up to 3-way interactions.
- n.cores : The number of CPU cores to use. The cross-validation loop will attempt to send different CV folds off to different cores.

Fitting the Model :

```
gbm_train <- data_process(JPL_train,2)
gbm_test <- data_process(JPL_test,2)
set.seed(131825431, sample.kind = "Rounding")
fit_gbm <- gbm(neo_class~., data = gbm_train, n.trees = 50,
               distribution = "multinomial", class.stratify.cv =TRUE,
               bag.fraction = 0.3, train.fraction = 0.7, shrinkage = 0.1,
               interaction.depth = 3, keep.data = FALSE, verbose = FALSE, n.cores = 6)
```

Visual representation of the model:

```
par(mfrow=c(1,1))
best.iter <- gbm.perf(fit_gbm, plot.it = FALSE)
p1 <- plot(fit_gbm)
p2 <- plot(fit_gbm, i.var = 1:2, n.trees = best.iter)
gridExtra::grid.arrange(p1,p2, ncol=2, nrow=1)
```



Prediction and the confusion matrix:

```
y_hat_gbm <- predict(fit_gbm, gbm_test, n.trees = best.iter, type='response')
labels <- colnames(y_hat_gbm)[apply(y_hat_gbm, 1, which.max)]

cm <- confusionMatrix(factor(labels), gbm_test$neo_class)
results <- bind_rows(results,results_process(as.data.frame(cm$byClass),"gbm"))
cm$table
```

```
##            Reference
## Prediction     NEO    PHO     SO
##         NEO  19030     15      0
##         PHO     34   1736      0
##         SO       0      0 841826
```

Results:

```
results %>% filter (model_name == "gbm")  %>% knitr::kable()
```

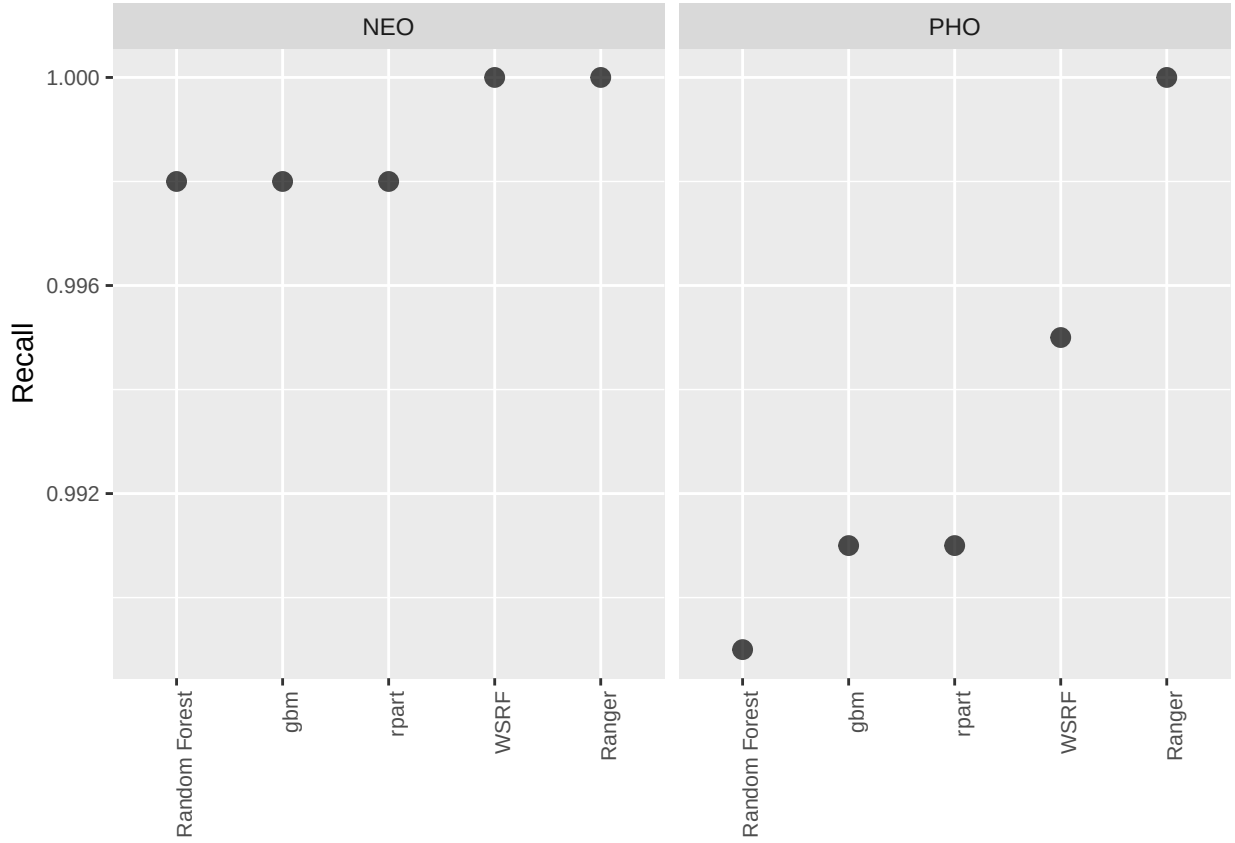
model_name	Object	Recall	F1
gbm	NEO	0.998	0.999
gbm	PHO	0.991	0.986
gbm	SO	1.000	1.000

- The misses on PHO and NEO increased.
- The Recall and F1 scores for PHO are 0.991 and 0.986

4.7 Collated Results

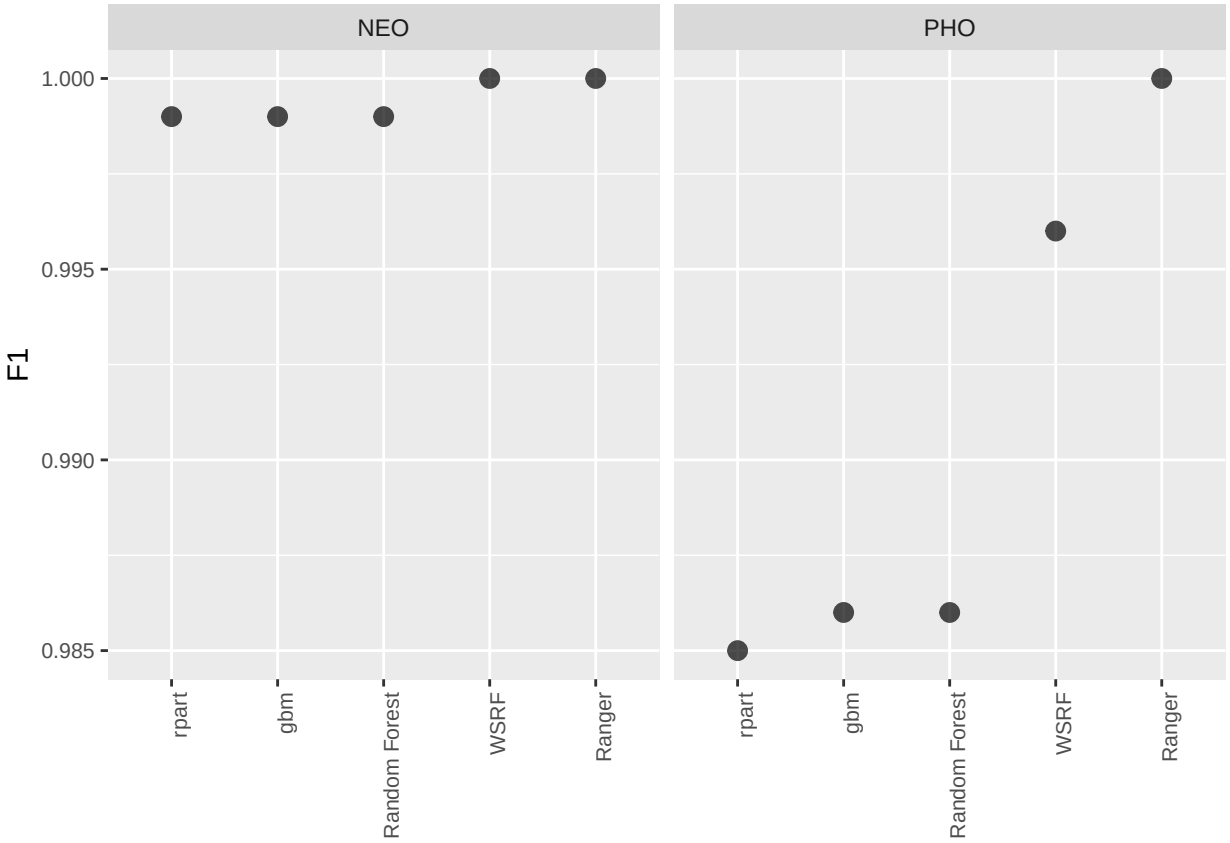
4.7.1 Recall

model_name	NEO	PHO	SO
gbm	0.998	0.991	1.000
Naive Bayes	0.985	0.692	0.995
PCA Multinom	0.052	0.035	1.000
Random Forest	0.998	0.989	1.000
Ranger	1.000	1.000	1.000
rpart	0.998	0.991	1.000
WSRF	1.000	0.995	1.000

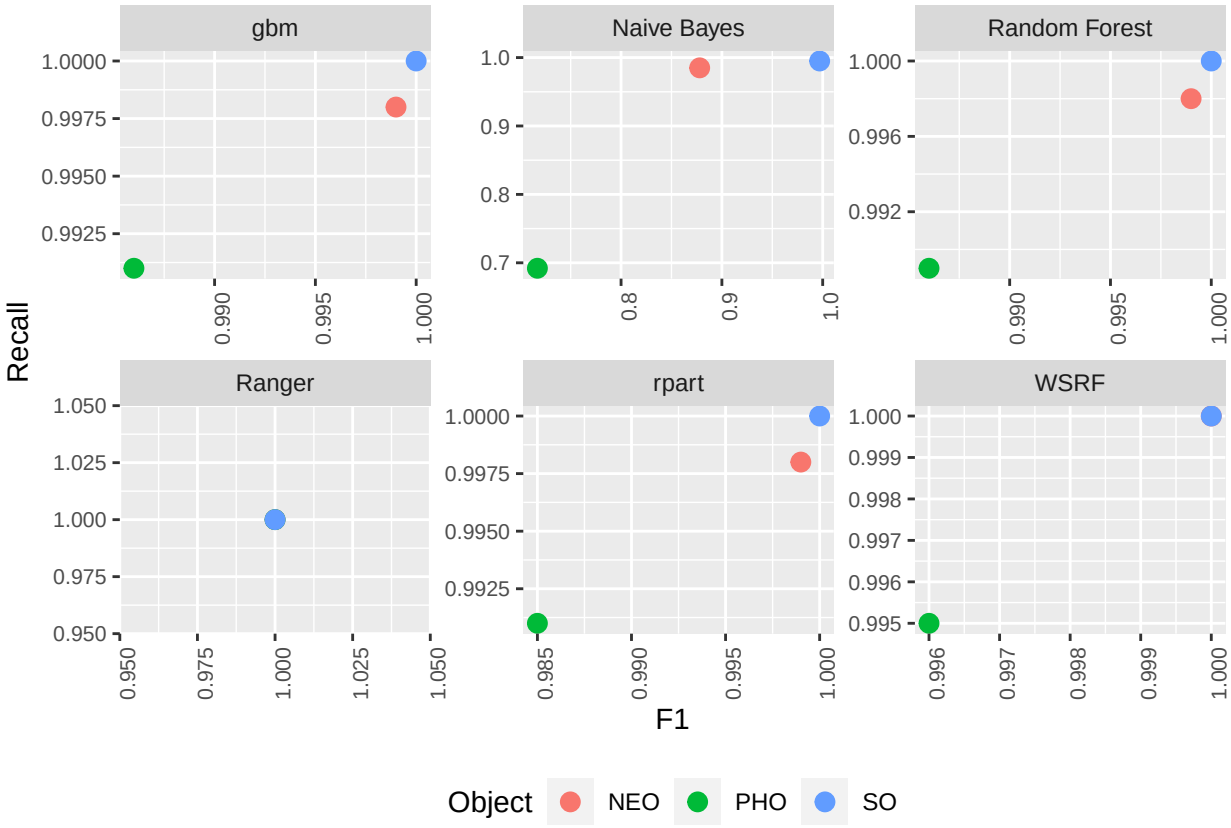


4.7.2 F1

model_name	NEO	PHO	SO
gbm	0.999	0.986	1.000
Naive Bayes	0.878	0.717	0.997
PCA Multinom	0.098	0.064	0.988
Random Forest	0.999	0.986	1.000
Ranger	1.000	1.000	1.000
rpart	0.999	0.985	1.000
WSRF	1.000	0.996	1.000



4.7.3 F1 vs Recall



- The Rpart, randomForest, wsrf, ranger and gbm models have a Recall > 0.99
- Based on the scores on the predictions from the training and test sets, the final validation models will be on Ranger and WSRF as these models have both the Recall and F1 ≥ 0.99 for all the three categories.
- Rpart is also tested on the validation set.

5 Final Models

5.1 Validation

5.1.1 Ranger

Validation set matrix:

```
#process as matrix for the model
validation <- data_process(JPL_validation,1)
JPL_F <- data_process(JPL_data,1)
```

Model Fitting, Prediction and the confusion matrix :

```
set.seed(131825431, sample.kind = "Rounding")
#fit the model for validation
final_ranger <- ranger(x=JPL_F$X, y=JPL_F$Y,
                       classification = TRUE, min.node.size = 1,
                       num.trees = rf_mtry$mtry*50, mtry=rf_mtry$mtry+1,
                       splitrule = 'gini', importance = 'impurity',
                       num.threads = 6, regularization.usedepth = TRUE,
                       verbose = FALSE, class.weights = c(7,11,0))

#predict and the confusion matrix
pred_ranger <- predict(final_ranger, validation$X)
cm <- confusionMatrix(pred_ranger$predictions, validation$Y)
results <- bind_rows(results,results_process(as.data.frame(cm$byClass),"Validation : Ranger"))
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO   2350      0      0
##        PHO      4    217      0
##        SO       0      0 103930
```

Results :

```
results %>% filter (model_name =="Validation : Ranger")  %>% knitr::kable()
```

model_name	Object	Recall	F1
Validation : Ranger	NEO	0.998	0.999
Validation : Ranger	PHO	1.000	0.991
Validation : Ranger	SO	1.000	1.000

- The Recall and the F1 for the Ranger model on the validation dataset, as validation is > 0.99

Inaccurate Predictions :

```
inaccurate_details <- rbind(inaccurate_details, inaccuracy_check(which(
  pred_ranger$predictions != validation$Y),'Validation : Ranger',JPL_validation))
inaccurate_details %>% filter (model_name == "Validation : Ranger") %>% .[,3:10] %>%
  tibble() %>% knitr::kable()
```

full_name	producer	neo	pha	per_y	moid	first_obs	last_obs
494975 (2009 WO106)	Otto Matic	1	0	1.6110	0.05000	2009-11-26	2018-01-13
(2009 QS)	Otto Matic	1	0	3.0708	0.04234	2009-08-17	2009-09-16
(2015 BW310)	Otto Matic	1	0	4.2435	0.02712	2014-07-31	2015-02-18
(2018 HA1)	Otto Matic	1	0	1.5291	0.02622	2018-04-19	2018-05-19

5.1.2 WSRF

Model Fitting, Prediction and the confusion matrix :

```
set.seed(131825431, sample.kind = "Rounding")
#fit the model for validation
final_wsrf <- wsrf(x=JPL_F$X, y=JPL_F$Y,
                   ntree = wsrf_mtry$mtry*100, mtry=wsrf_mtry$mtry+1,
                   weights=TRUE,  importance = TRUE, parallel = TRUE)
#predict and the confusion matrix
pred_wsrf <- predict(final_wsrf, validation$X)
pred_wsrf_m <- as.matrix(pred_wsrf$class)
pred_wsrf_m <- ifelse(pred_wsrf_m == 1, 'NEO',ifelse(pred_wsrf_m == 2, 'PHO','SO'))

cm <- confusionMatrix(as.factor(pred_wsrf_m),validation$Y)
results <- bind_rows(results,results_process(as.data.frame(cm$byClass),"Validation : WSRF"))
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO   2351      0      0
##        PHO      3    217      0
##        SO       0      0 103930
```

Results :

```
results %>% filter (model_name == "Validation : WSRF")  %>% knitr::kable()
```

model_name	Object	Recall	F1
Validation : WSRF	NEO	0.999	0.999
Validation : WSRF	PHO	1.000	0.993
Validation : WSRF	SO	1.000	1.000

- The Recall for the PHO and SO is 1, with that of NEO > 0.99
- The F1 for all the ctegories is > 0.99

Inaccurate Predictions :

```
inaccurate_details <- rbind(inaccurate_details, inaccuracy_check(which(
  validation$Y != pred_wsrf_m),"Validation : WSRF",JPL_validation))
inaccurate_details %>% filter (model_name == "Validation : WSRF") %>% .[,3:10] %>%
  tibble() %>% knitr::kable()
```

full_name	producer	neo	pha	per_y	moid	first_obs	last_obs
(2009 QS)	Otto Matic	1	0	3.0708	0.04234	2009-08-17	2009-09-16
(2015 BW310)	Otto Matic	1	0	4.2435	0.02712	2014-07-31	2015-02-18
(2018 HA1)	Otto Matic	1	0	1.5291	0.02622	2018-04-19	2018-05-19

5.1.3 Rpart

Model Fitting, Prediction and the confusion matrix :

```
#process as data tables for the model
validation <- data_process(JPL_validation,2)
JPL_F <- data_process(JPL_data,2)
#fit model
Final_rpart <- rpart(neo_class ~. , data=JPL_F,
             control = rpart.control(cp = rpart_cp, minsplit = 10, minbucket=1))
#predict and confusion matrix
pred_rpart <- predict(Final_rpart, validation)
labels <- colnames(pred_rpart)[apply(pred_rpart, 1, which.max)]
cm <- confusionMatrix(factor(labels), validation$neo_class)
results <- bind_rows(results,results_process(as.data.frame(cm$byClass),"Validation : rpart"))
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO   2350      0      0
##        PHO      4    217      0
##        SO       0      0 103930
```

Results :

```
results %>% filter (model_name == "Validation : rpart")  %>% knitr::kable()
```

model_name	Object	Recall	F1
Validation : rpart	NEO	0.998	0.999
Validation : rpart	PHO	1.000	0.991
Validation : rpart	SO	1.000	1.000

- The Recall and the F1 for all the categories is > 0.99

Inaccurate Predictions :

```
inaccurate_details <- rbind(inaccurate_details, inaccuracy_check(which(
  labels != validation$neo_class),"Validation : rpart",JPL_validation))
inaccurate_details %>% filter (model_name == "Validation : rpart") %>% .[,3:10] %>%
  tibble() %>% knitr::kable()
```

full_name	producer	neo	pha	per_y	moid	first_obs	last_obs
494975 (2009 WO106)	Otto Matic	1	0	1.6110	0.05000	2009-11-26	2018-01-13
(2009 QS)	Otto Matic	1	0	3.0708	0.04234	2009-08-17	2009-09-16
(2015 BW310)	Otto Matic	1	0	4.2435	0.02712	2014-07-31	2015-02-18
(2018 HA1)	Otto Matic	1	0	1.5291	0.02622	2018-04-19	2018-05-19

5.2 As Holdout

5.2.1 WSRF

- Illustration : Using the fit from the train set and treating the validation data as a holdout data.

Validation set matrix:

```
#process as matrix for the model
validation <- data_process(JPL_validation,1)
JPL_F <- data_process(JPL_data,1)
```

Prediction and the confusion matrix :

```
y_f_wsrf <- predict(fit_wsrf, validation$X)
y_f_wsrf_m <- as.matrix(y_f_wsrf$class)
y_f_wsrf_m <- ifelse(y_f_wsrf_m == 1, 'NEO',ifelse(y_f_wsrf_m == 2, 'PHO','SO'))
cm <- confusionMatrix(as.factor(y_f_wsrf_m),validation$Y)
results <- bind_rows(results,results_process(as.data.frame(cm$byClass),"Holdout : WSRF"))
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO   2351      0      0
##        PHO      3    217      0
##        SO       0      0 103930
```

Results :

```
results %>% filter (model_name == "Holdout : WSRF")  %>% knitr::kable()
```

model_name	Object	Recall	F1
Holdout : WSRF	NEO	0.999	0.999
Holdout : WSRF	PHO	1.000	0.993
Holdout : WSRF	SO	1.000	1.000

Inaccurate Predictions :

```
inaccurate_details <- rbind(inaccurate_details, inaccuracy_check(which(
  y_f_wsrf_m != validation$Y ),'Holdout : WSRF',JPL_validation))
inaccurate_details %>% filter (model_name == "Holdout : WSRF") %>% .[,3:10] %>%
  tibble() %>% knitr::kable()
```

full_name	producer	neo	pha	per_y	moid	first_obs	last_obs
(2009 QS)	Otto Matic	1	0	3.0708	0.04234	2009-08-17	2009-09-16
(2015 BW310)	Otto Matic	1	0	4.2435	0.02712	2014-07-31	2015-02-18
(2018 HA1)	Otto Matic	1	0	1.5291	0.02622	2018-04-19	2018-05-19

5.2.2 Ranger

- Illustration : Using the fit from the train set and treating the validation data as a holdout data.

Prediction and the confusion matrix :

```
set.seed(131825431, sample.kind = "Rounding")
#predict using the training fitted model against the validation set.
pred_O_ranger <- predict(fit_ranger, validation$X)
cm <- confusionMatrix(pred_O_ranger$predictions, validation$Y)
results <- bind_rows(results,results_process(as.data.frame(cm$byClass),"Holdout : Ranger"))
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO   2350      0      0
##        PHO      4    217      0
##        SO       0      0 103930
```

Results :

```
results %>% filter (model_name =="Holdout : Ranger")  %>% knitr::kable()
```

model_name	Object	Recall	F1
Holdout : Ranger	NEO	0.998	0.999
Holdout : Ranger	PHO	1.000	0.991
Holdout : Ranger	SO	1.000	1.000

- The Recall and the F1 for the Ranger model on the validation dataset, as a holdout is > 0.99

Inaccurate Predictions :

```
inaccurate_details <- rbind(inaccurate_details, inaccuracy_check(which(
  pred_O_ranger$predictions != validation$Y ),'Holdout : Ranger',JPL_validation))
inaccurate_details %>% filter (model_name == "Holdout : Ranger") %>% .[,3:10] %>%
  tibble() %>% knitr::kable()
```

full_name	producer	neo	pha	per_y	moid	first_obs	last_obs
494975 (2009 WO106)	Otto Matic	1	0	1.6110	0.05000	2009-11-26	2018-01-13
(2009 QS)	Otto Matic	1	0	3.0708	0.04234	2009-08-17	2009-09-16
(2015 BW310)	Otto Matic	1	0	4.2435	0.02712	2014-07-31	2015-02-18
(2018 HA1)	Otto Matic	1	0	1.5291	0.02622	2018-04-19	2018-05-19

5.2.3 Gradient Boost

- Illustration : Using the fit from the train set and treating the validation data as a holdout data.

Validation set data table:

```
#process as matrix for the model
validation <- data_process(JPL_validation,2)
```

Prediction and the confusion matrix:

```
y_f_gbm <- predict(fit_gbm, validation, n.trees = best.iter, type='response')
labels <- colnames(y_f_gbm)[apply(y_f_gbm, 1, which.max)]
cm <- confusionMatrix(factor(labels), validation$neo_class)
results <- bind_rows(results,results_process(as.data.frame(cm$byClass),"Holdout : gbm"))
cm$table
```

```
##           Reference
## Prediction    NEO    PHO     SO
##        NEO   2351      0      0
##        PHO      3    217      0
##        SO       0      0 103930
```

Results :

```
results %>% filter (model_name == "Holdout : gbm")  %>% knitr::kable()
```

model_name	Object	Recall	F1
Holdout : gbm	NEO	0.999	0.999
Holdout : gbm	PHO	1.000	0.993
Holdout : gbm	SO	1.000	1.000

Inaccurate Predictions :

```
inaccurate_details <- rbind(inaccurate_details, inaccuracy_check(which(
  labels != validation$neo_class),'Holdout : gbm',JPL_validation))
inaccurate_details %>% filter (model_name == "Holdout : gbm") %>% .[,3:10] %>%
  tibble() %>% knitr::kable()
```

full_name	producer	neo	pha	per_y	moid	first_obs	last_obs
(2009 QS)	Otto Matic	1	0	3.0708	0.04234	2009-08-17	2009-09-16
(2015 BW310)	Otto Matic	1	0	4.2435	0.02712	2014-07-31	2015-02-18
(2018 HA1)	Otto Matic	1	0	1.5291	0.02622	2018-04-19	2018-05-19

6 Conclusion

The JPL small body dataset was downloaded from from the **https://ssd.jpl.nasa.gov/sbdb_query.cgi** website, cleaned and initial variable assessment is done using various techniques.

Additional derived variables were also included for model evaluations. t-tests to check the statistical significance of the variations in each variable was also done separately (*summary included in R code file*). However, the details have not been included here in order to maintain concision of the case study document.

In the model evaluation the multinom (PCA), Naive Bayes, rpart, randomForest, wsrf, ranger and gbm models were eveluated. An ensemble of lda, qda, knn, rpart, svmRadialCost and wsrf was used for direction for a suitable model.

The classification of Small Bodies model was built using only 9 variables $H, q, moid, m\_d, e, t\_jup, a, i, T\_p$, of which 2 variables m_d *and* T_p are derived variables.

The Final validation was done with The Ranger model (a fast implementation of random forests), WSRF (Forest of Weighted Subspace Decision Trees) and rpart (Recursive Partitioning and Regression Trees).

The Final Scores:

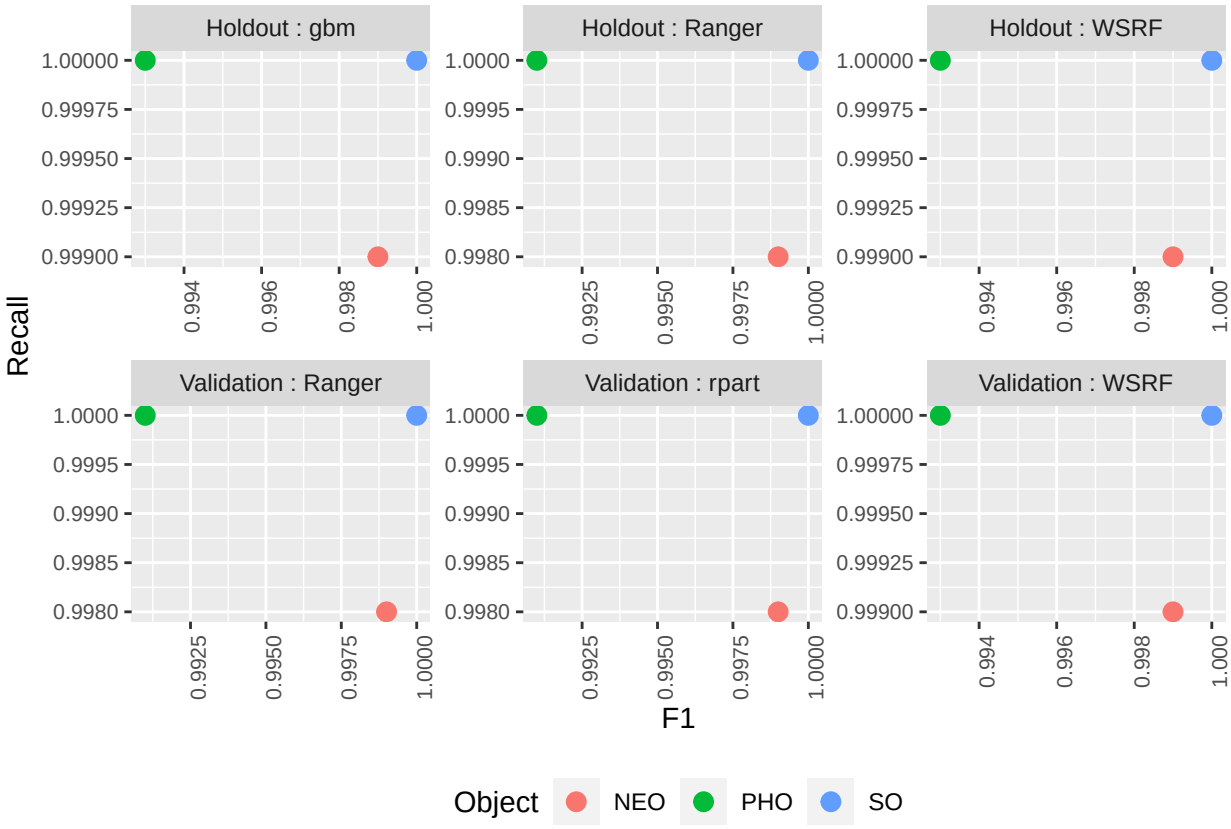
Recall

model_name	NEO	PHO	SO
Holdout : gbm	0.999	1	1
Holdout : Ranger	0.998	1	1
Holdout : WSRF	0.999	1	1
Validation : Ranger	0.998	1	1
Validation : rpart	0.998	1	1
Validation : WSRF	0.999	1	1

F1

model_name	NEO	PHO	SO
Holdout : gbm	0.999	0.993	1
Holdout : Ranger	0.999	0.991	1
Holdout : WSRF	0.999	0.993	1
Validation : Ranger	0.999	0.991	1
Validation : rpart	0.999	0.991	1
Validation : WSRF	0.999	0.993	1

The Objective of the case study, to build a machine learning model to predict the classification of the small solar bodies into PHO, NEO and SO (other small solar bodies), with the target measures of Recall for each category above 0.99 and F1 Score for each category above 0.99 was achieved.



Further tweaking and tuning of the models and reassessing variables, both observed and derived might achieve an absolute Recall and F1 score of 1.

6.1 References

- **Rafael A. Irizarry** Introduction to Data Science: **https://rafalab.github.io/dsbook/**
- NASA JPL website **https://ssd.jpl.nasa.gov/**
- Articles on https://wikipedia.org for the brief history details and the two images.

veneet.bhardwaj@gmail.com

6.2 Annexure

Recall and F1 Scores of each model:

```
results %>% knitr::kable()
```

model_name	Object	Recall	F1
PCA Multinom	NEO	0.052	0.098
PCA Multinom	PHO	0.035	0.064
PCA Multinom	SO	1.000	0.988
Naive Bayes	NEO	0.985	0.878
Naive Bayes	PHO	0.692	0.717
Naive Bayes	SO	0.995	0.997
rpart	NEO	0.998	0.999
rpart	PHO	0.991	0.985
rpart	SO	1.000	1.000
Random Forest	NEO	0.998	0.999
Random Forest	PHO	0.989	0.986
Random Forest	SO	1.000	1.000
WSRF	NEO	1.000	1.000
WSRF	PHO	0.995	0.996
WSRF	SO	1.000	1.000
Ranger	NEO	1.000	1.000
Ranger	PHO	1.000	1.000
Ranger	SO	1.000	1.000
gbm	NEO	0.998	0.999
gbm	PHO	0.991	0.986
gbm	SO	1.000	1.000
Validation : Ranger	NEO	0.998	0.999
Validation : Ranger	PHO	1.000	0.991
Validation : Ranger	SO	1.000	1.000
Validation : WSRF	NEO	0.999	0.999
Validation : WSRF	PHO	1.000	0.993
Validation : WSRF	SO	1.000	1.000
Validation : rpart	NEO	0.998	0.999
Validation : rpart	PHO	1.000	0.991
Validation : rpart	SO	1.000	1.000
Holdout : WSRF	NEO	0.999	0.999
Holdout : WSRF	PHO	1.000	0.993
Holdout : WSRF	SO	1.000	1.000
Holdout : Ranger	NEO	0.998	0.999
Holdout : Ranger	PHO	1.000	0.991
Holdout : Ranger	SO	1.000	1.000
Holdout : gbm	NEO	0.999	0.999
Holdout : gbm	PHO	1.000	0.993
Holdout : gbm	SO	1.000	1.000

veneet.bhardwaj@gmail.com