<u>Detection of Habitable Planets suitable for Space Colonization</u>

Abstract

From the center of the Earth to the far-flung galaxies, we find evidence that life arose from cosmic processes. The iron in our blood and the calcium in our bones were made inside stars. All silver and gold were forged by stars that exploded long ago. Terrestrial life is embedded in a cosmic web, and it seems reasonable to speculate that life is cosmically commonplace. If life is in fact cosmically commonplace, where might we find it? Our search begins **within the solar system**, as we try to locate three ingredients upon which life depends: water, energy, and organic molecules (or carbon). Recent discoveries inform us that these requisites may exist well beyond the planets closely orbiting the sun. This area where conditions might potentially support life is called **The Habitable Zone.**

<u>Previous Definition</u>: The habitable zone first encompassed the orbits of Venus to Mars, planets close enough to the sun for solar energy to drive the chemistry of life — but not so close as to boil off water or break down the organic molecules on which life depends.	**<u>Expanding Definition</u>**: But the habitable zone may be larger than originally conceived. The strong gravitational pull caused by large planets may produce enough energy to sufficiently heat the cores of orbiting moons. Life has proven itself tough here on Earth. Perhaps it could thrive in more extreme environments

Now, the searching of the habitable planets are based on the following parameters-

1. **E.S.I.** - The **Earth Similarity Index** (ESI) is a proposed characterization of how similar a planetary-mass object or natural satellite is to **Earth**. It was designed to be a scale from 0-1, with **Earth** having a value of 1. According to this measure, as of 23 July 2015, the confirmed planet currently thought to be most similar to Earth on **mass**, **radius** and **temperature** is **Kepler**-438b. Scientists estimate that there may be billions of Earth-size planets within the **Milky Way** galaxy alone.
2. Luminosity Calculation-

Motivation

At the age of 8, the idea of aliens first came across my mind from Satyajit Ray's short story, **Bonku Babur Bondhu.**

When I was 10, the Jules Verne's classics-
Around The World In Eighty Days
From The Earth To The Moon and
Journey To The Centre Of The Earth gave me food for my imagination about extraterrestrial life.

Eventually, once I got the chance to go through the famous sci-fi magazine, **Imagination.**

Being a voracious Marvel Comics reader, I ended up watching **Guardians Of The Galaxy** and by the time I've already become a big fan of **Star Wars**, an American epic space opera franchise, created by George Lucas.

In the last three years the books which completely got me awestruck and compelled me to do some basic studies and research on the topic that my project is based on, are the following masterpieces by Stephen Hawking-
The Universe In A Nutshell
The Theory Of Everything
The Future Of Spacetime
Black Holes: The Reith Lectures

Synopsis

1.The softwares I used -

Jupyter Notebook
Spyder IDE

2.The languages I used -

Python 3

Python is a clear and powerful object-oriented programming language, comparable to Perl, Ruby, Scheme, or Java. **Some of Python's notable features**:

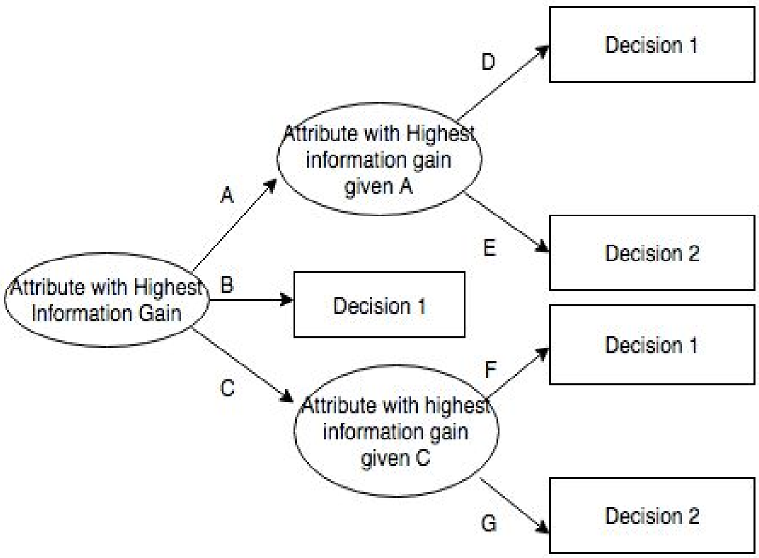
- Uses an elegant syntax, making programs easier to read.
- Python is ideal for prototype development without compromising maintainability.
- Provides large standard library supporting common programming task i.e connecting to web servers, searching text with regular expressions, reading, modifying files.
- Python's interactive mode makes it easy to test short snippets of code. There's also a bundled development environment called IDLE.
- Extended by adding new modules implemented in a compiled language i.e **C /C++.**
- Runs anywhere, including **Mac OS X, Windows, Linux, and Unix, Android and iOS**.
- Is free software in two senses. It doesn't cost anything to download or use Python or to include it in an application. Python can also be freely modified and re-distributed because while the language is copyrighted it's available under an open source license.

Some programming-language features of Python are:

- *A variety of basic data types are available:* **numbers** *(floating point, complex, and unlimited-length long integers),* **strings** *(both ASCII and Unicode),* **lists**, **dictionaries**.
- *Python supports object-oriented programming with classes and multiple inheritances.*
- *Codes can be grouped into modules and packages.*
- *Supports raising and catching exceptions, resulting in cleaner error handling.*
- *Data types are strongly and dynamically typed. Mixing incompatible types (attempt to add a string and a number) causes an exception to be raised, so errors are caught sooner.*
- *Contains advanced programming features (***generators*** and ***list comprehensions***.)*
- *Automatic memory management frees from manual allocation and free memory in code.*

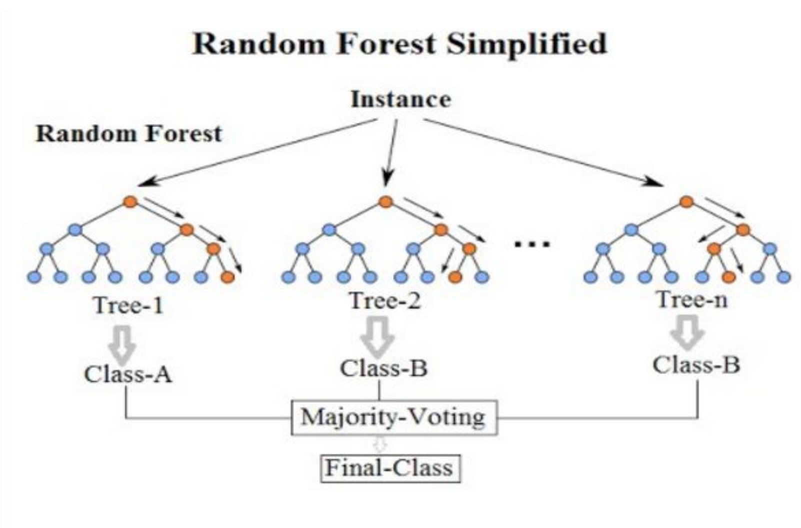
3.Algorithms I used -

Decision tree algorithm- A decision tree is a flow-chart-like structure where each internal (non-leaf) node denotes a test on an attribute, each branch represents the outcome of a test, and each leaf (or terminal) node holds a class label. The topmost node in a tree is the root node. There are many specific decision-tree algorithms-



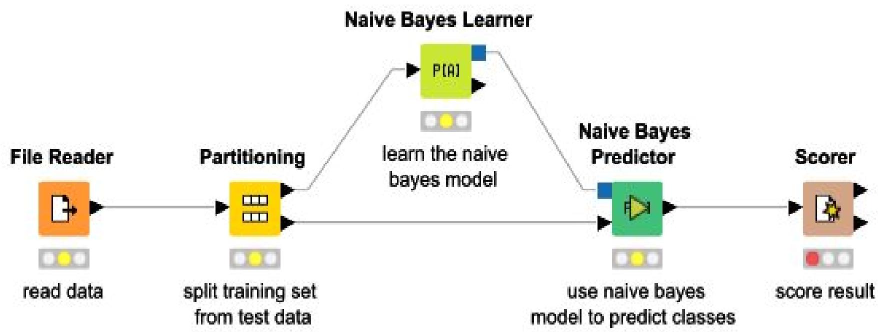
ID3 (Iterative Dichotomiser 3)
C4.5 (successor of ID3)
CART (Classification And Regression Tree)
CHAID (Chi-squared Automatic Interaction Detector) Performs multi-level splits.
MARS: extends decision trees to handle numerical data better.

Random forest Algorithm -

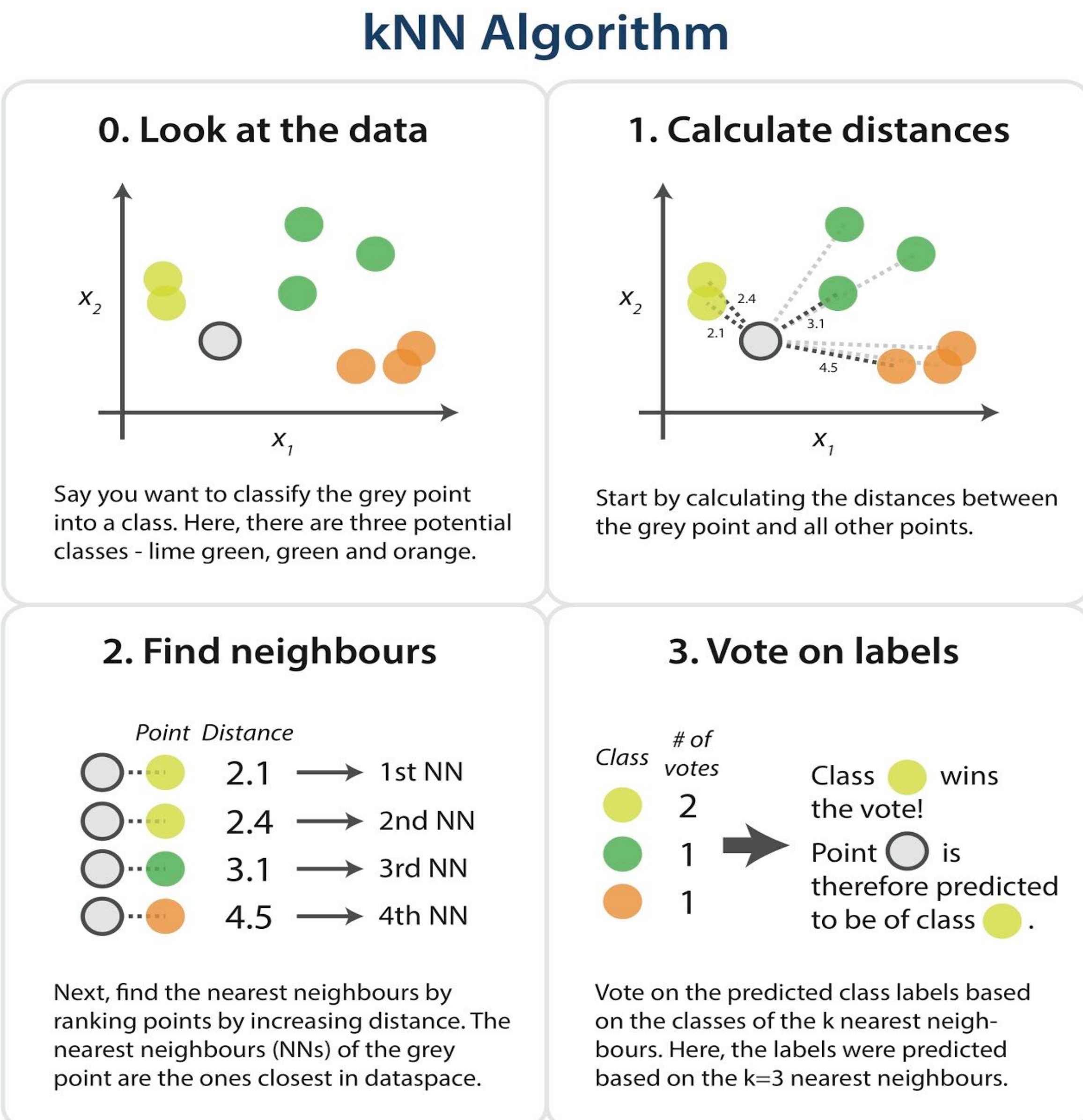


Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

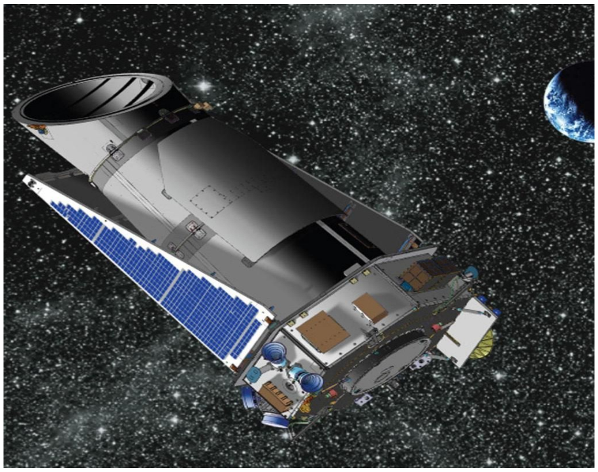
Naive Bayes Algorithm - Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.



K-nearest Neighbours Algorithm:

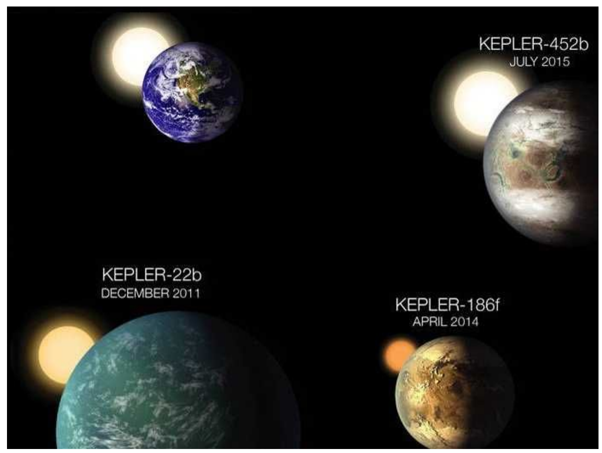


4. Basic Information



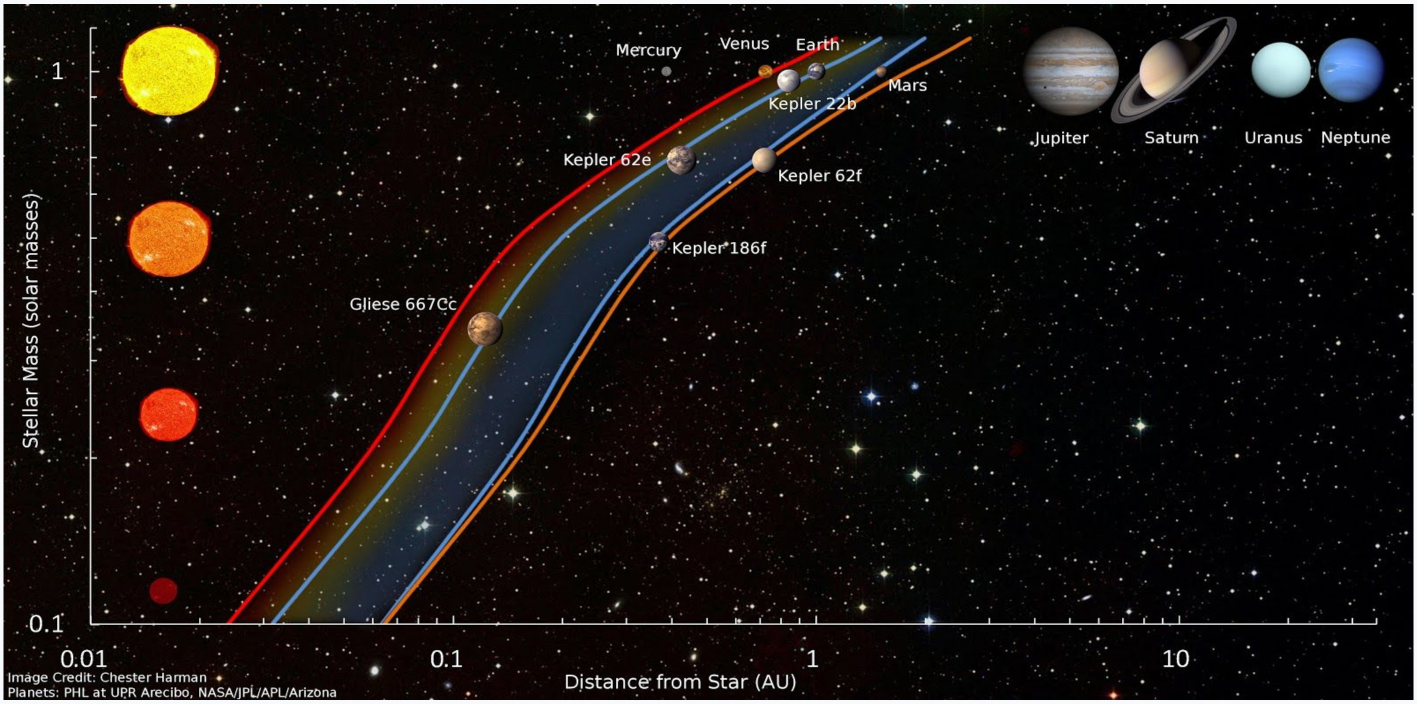
Kepler space telescope has discovered 70 per cent of the 3,800 confirmed alien world to date, has run out of fuel, agency officials announced on Oct 30, 2018. Kepler can no longer reorient itself to study cosmic objects or beam its data to Earth, so the legendary instrument's in-space work is done after nearly a decade. It was a space observatory launched by NASA to discover Earth-size planets orbiting other stars. Named after astronomer Johannes Kepler, the spacecraft was launched on March 7, 2009, into an Earth-trailing heliocentric orbit.

Super Earth : A super-Earth or mini-Neptune is an extrasolar planet with a mass higher than Earth's, but substantially below those of the Solar System's ice giants, Uranus and Neptune, which are 15 and 17 times Earth's, respectively. The term "super-Earth" refers only to the mass of the planet implying nothing about the surface conditions, i.e habitability. The alternative term "gas dwarfs" may be more accurate for those at the higher end of the mass scale.



Hot Earth: NASA's Transiting Exoplanet Survey Satellite **(TESS)** made an early discovery of "super-Earth" and "hot Earth" planets in solar systems at least 49 light-years away. TESS announced the first candidate planet of the mission would be a Super-Earth, almost 60 light years away from Earth the planet orbits around the bright star Pi Mensae once every 6.3 days. The planet's mass and radius reportedly show a **water-like density** and is its system's second known planet. The other planet orbiting Pi Mensae, Pi Mensae b, was discovered in 2001 and has a mass 10 times the size of Jupiter. It orbits the star every 5.7 years. This planet is slightly larger than the earth and is slightly closer at only 49 light-years away. It orbits the M dwarf star, LHS 3844, and each rotation takes only 11 hours

Circumstellar habitable zone: In astronomy, the habitable zone is the range of orbits around a star within which a planetary surface can support liquid water given sufficient atmospheric pressure.



Searching of habitable alternative planets suitable for Space Colonization

Ooops, It's **not Earth**. It's *Gliese 581g* or you may call it as ***Zarmina***

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import matplotlib.colors as clr
import matplotlib.cm as cm
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns

from scipy.stats.stats import pearsonr
```

Requirements Specification and analysis

The dataset(PHL's Exoplanet Catalog of the Planetary Habitability Laboratory) I've used is created and maintained by

The Planetary Habitability Laboratory, University of Puerto Rico at Arecibo

It can be found at the following url: http://phl.upr.edu/projects/habitable-exoplanets-catalog/data/database (http://phl.upr.edu/projects/habitable-exoplanets-catalog/data/database)

The PHL's Exoplanets Catalog (PHL-EC) contains observed and modeled parameters for all currently confirmed exoplanets from the Extrasolar Pl anets Encyclopedia and NASA Kepler candidates from the NASA Exoplanet Archive, including those potentially habitable. It also contains a few still unconfirmed exoplanets of interest. The main difference between PHL-EC and other exoplanets databases is that it contains more estimated stellar and planetary parameters, habitability assessments with various habitability metrics, planetary classifications, and many corrections. Some interesting inclusions are the identification of those stars in the Catalog of Nearby Habitable Systems (HabCat aka HabStar Catalog), the apparent size and brightness of stars and planets as seen from a vantage point (i.e. moon-Earth distance), and the location constellation of each planet.

Reasons behind chosing this dataset over any other is because of it's expanded target list combining measures and modeled parameters from various sources. Hence, it provides a good metric for visualization and statistical analysis.

PHL-EC consists of 68 features and 3875 confirmed exoplanets.

In [2]:
```
allExoplanets = pd.read_csv('confirmed_exoplanets.csv',low_memory=False)

print('Features, Data Points = '+str(allExoplanets.shape))
```

```
Features, Data Points = (3875, 68)
```

In [3]:
```
print('All Features of PHL-EC:\n\n')
for i in allExoplanets:
    print("{feature}".format(feature=i),sep='\t')
```

```
All Features of PHL-EC:


P. Name
P. Name Kepler
P. Name KOI
P. Zone Class
P. Mass Class
P. Composition Class
P. Atmosphere Class
P. Habitable Class
P. Min Mass (EU)
P. Mass (EU)
P. Max Mass (EU)
P. Radius (EU)
P. Density (EU)
P. Gravity (EU)
P. Esc Vel (EU)
P. SFlux Min (EU)
P. SFlux Mean (EU)
P. SFlux Max (EU)
P. Teq Min (K)
P. Teq Mean (K)
P. Teq Max (K)
P. Ts Min (K)
P. Ts Mean (K)
P. Ts Max (K)
P. Surf Press (EU)
P. Mag
P. Appar Size (deg)
P. Period (days)
P. Sem Major Axis (AU)
P. Eccentricity
P. Mean Distance (AU)
P. Inclination (deg)
P. Omega (deg)
S. Name
S. Name HD
S. Name HIP
S. Constellation
S. Type
S. Mass (SU)
S. Radius (SU)
S. Teff (K)
S. Luminosity (SU)
S. [Fe/H]
S. Age (Gyrs)
S. Appar Mag
S. Distance (pc)
S. RA (hrs)
S. DEC (deg)
S. Mag from Planet
S. Size from Planet (deg)
S. No. Planets
S. No. Planets HZ
S. Hab Zone Min (AU)
S. Hab Zone Max (AU)
P. HZD
P. HZC
P. HZA
P. HZI
P. SPH
P. Int ESI
P. Surf ESI
P. ESI
S. HabCat
P. Habitable
P. Hab Moon
P. Confirmed
P. Disc. Method
P. Disc. Year
```

In [4]:
```
## object type scienctific notation ke float korar portion ## last checkpoint

allExoplanets['P. SFlux Max (EU)'] = pd.to_numeric(allExoplanets['P. SFlux Max (EU)'],errors='coerce')

allExoplanets['P. SFlux Mean (EU)'] = pd.to_numeric(allExoplanets['P. SFlux Mean (EU)'],errors='coerce')

allExoplanets['P. SFlux Min (EU)'] = pd.to_numeric(allExoplanets['P. SFlux Min (EU)'],errors='coerce')
```

In [5]:
```
cat = len(allExoplanets.select_dtypes(include=['object']).columns)
num = len(allExoplanets.select_dtypes(include=['int64','float64']).columns)
print('Features of allExoplanets consists of ', cat, 'categorical', ' and ',
      num, 'numerical features')
print('\n\nCategorical Features:\n ')
for i in allExoplanets.select_dtypes(include=['object']).columns:
    print("{feature}".format(feature=i),end='\t')
print('\n\nNumerical Features:\n ')
for i in allExoplanets.select_dtypes(include=['int64','float64']).columns:
    print("{feature}".format(feature=i),end='\t')
```

```
Features of allExoplanets consists of  14 categorical  and  54 numerical features


Categorical Features:

P. Name P. Name Kepler  P. Zone Class   P. Mass Class   P. Composition Class    P. Atmosphere Class     P. Ha
bitable Class   S. Name S. Name HD      S. Name HIP     S. Constellation        S. Type P. Disc. Method P. Di
sc. Year

Numerical Features:

P. Name KOI     P. Min Mass (EU)        P. Mass (EU)    P. Max Mass (EU)        P. Radius (EU)  P. Density (E
U)      P. Gravity (EU) P. Esc Vel (EU) P. SFlux Min (EU)       P. SFlux Mean (EU)      P. SFlux Max (EU)
        P. Teq Min (K)  P. Teq Mean (K) P. Teq Max (K)  P. Ts Min (K)   P. Ts Mean (K)  P. Ts Max (K)   P. Su
rf Press (EU)   P. Mag  P. Appar Size (deg)     P. Period (days)        P. Sem Major Axis (AU)  P. Eccentrici
ty      P. Mean Distance (AU)   P. Inclination (deg)    P. Omega (deg)  S. Mass (SU)    S. Radius (SU)  S. Te
ff (K)  S. Luminosity (SU)      S. [Fe/H]       S. Age (Gyrs)   S. Appar Mag    S. Distance (pc)        S. RA
(hrs)   S. DEC (deg)    S. Mag from Planet      S. Size from Planet (deg)       S. No. Planets  S. No. Planet
s HZ    S. Hab Zone Min (AU)    S. Hab Zone Max (AU)    P. HZD  P. HZC  P. HZA  P. HZI  P. SPH  P. Int ESI
        P. Surf ESI     P. ESI  S. HabCat       P. Habitable    P. Hab Moon     P. Confirmed
```

correction:

P. Habitable is also a Categorical Variable, since it has only two unique values 0,1 indicates Yes and No respectively

In [6]:
```
desc = pd.DataFrame()
for c in allExoplanets:
    desc[c]=(allExoplanets[c].describe())
desc.head()
```

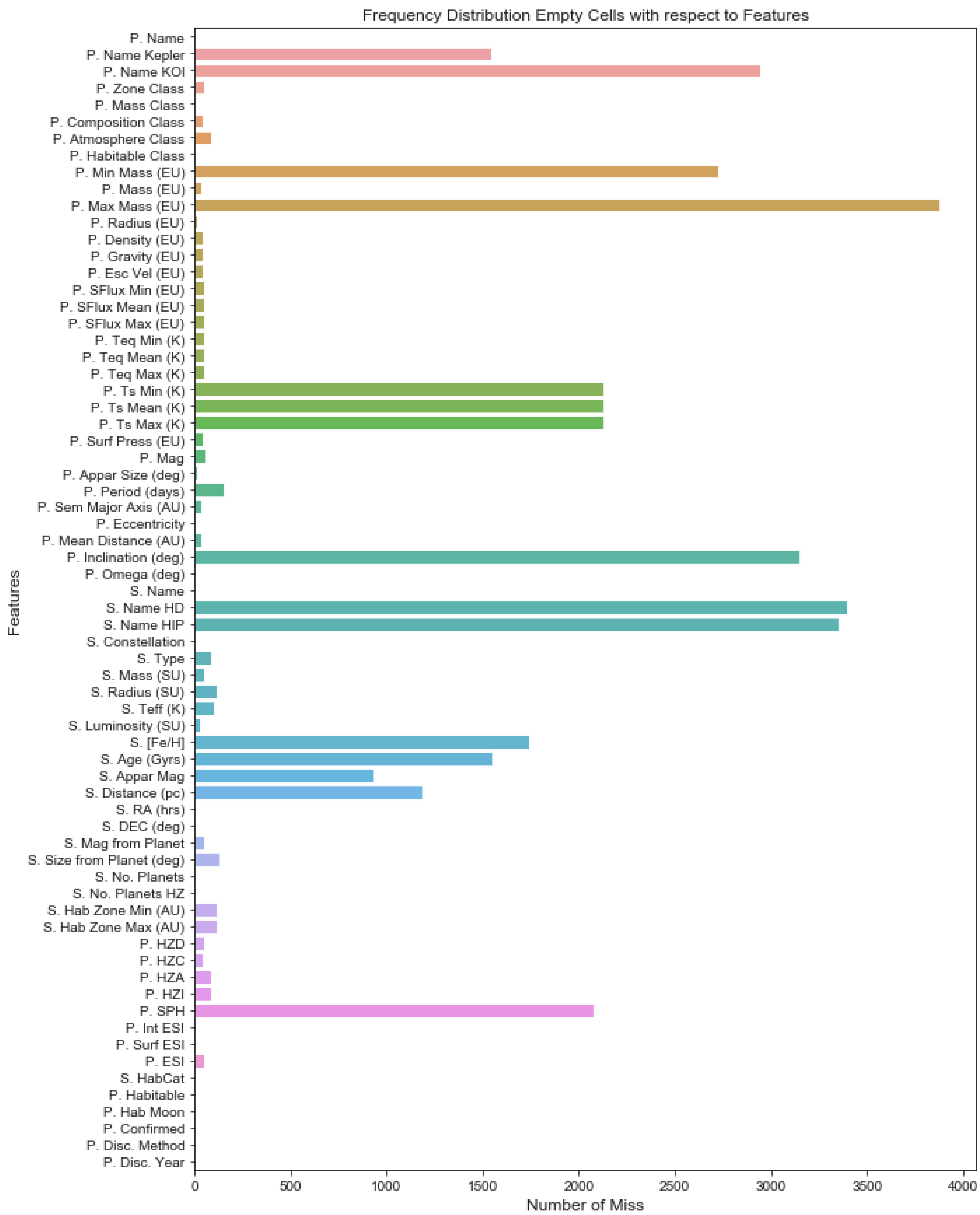
Out[6]:

	P. Name	P. Name Kepler	P. Name KOI	P. Zone Class	P. Mass Class	P. Composition Class	P. Atmosphere Class	P. Habitable Class	P. Min Mass (EU)	P. Mass (EU)	...	P. SPH	P. Int ESI	P. Surf ESI
count	3875	2328	933.0	3829	3869	3834	3790	3875	1148.0	3842.0	...	1801.0	3875.0	3875.0
unique	3875	2328	NaN	3	6	5	3	5	NaN	NaN	...	NaN	NaN	NaN
top	HD 30177 c	Kepler-1410 b	NaN	Hot	Jovian	gas	metals-rich	non-habitable	NaN	NaN	...	NaN	NaN	NaN
freq	1	1	NaN	3251	1273	2074	2372	3820	NaN	NaN	...	NaN	NaN	NaN

4 rows × 68 columns

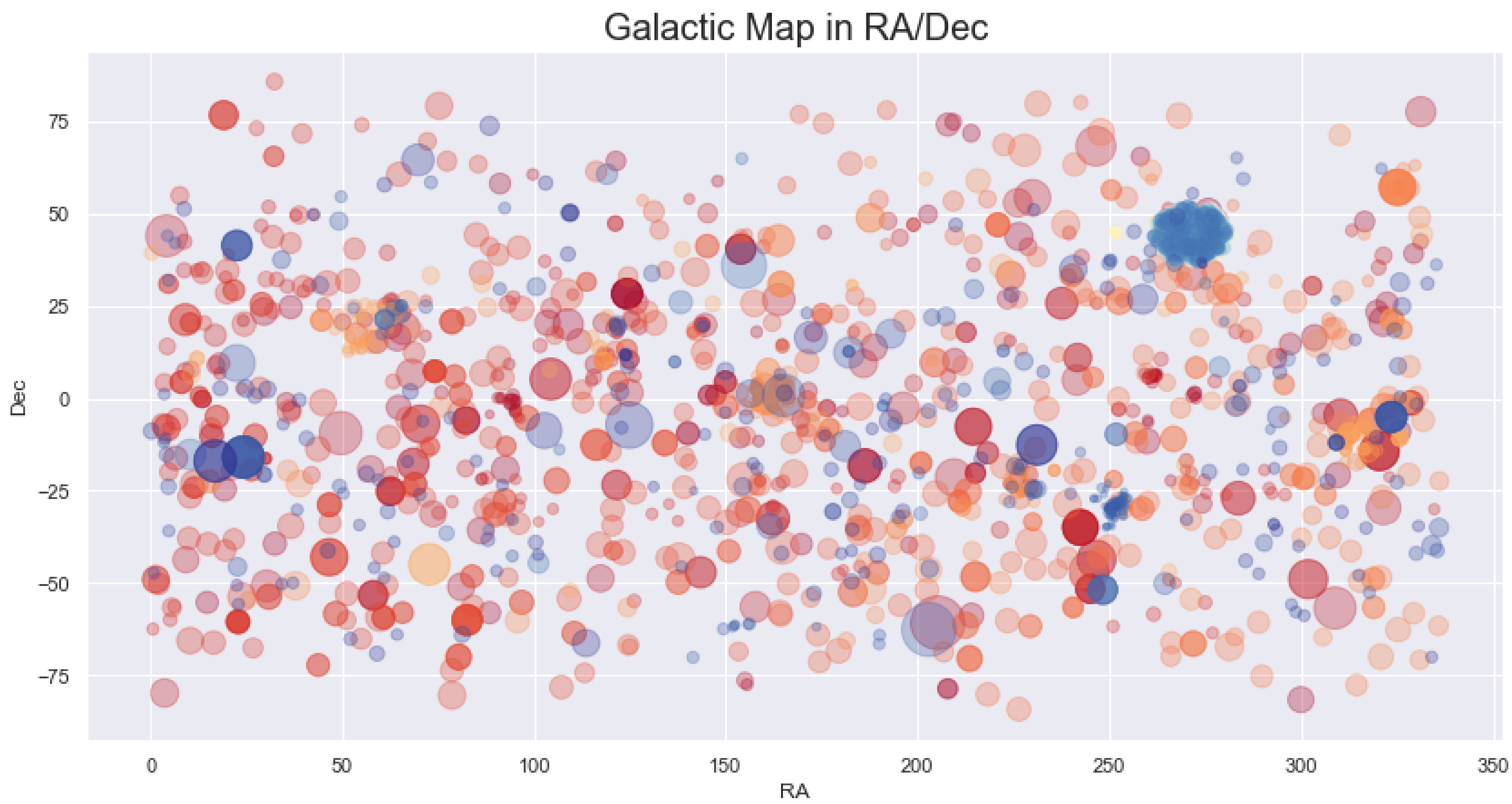
Percentage of Null Value in each feature of the given data set

In [7]:
```
#allExoplanets.isnull()
###print(allExoplanets.isnull().sum() * 100 / len(allExoplanets),end='\t\t\t\t')
f, ax = plt.subplots(figsize=(10, 15))

stat_count = allExoplanets.isnull().sum()
sns.set(style="darkgrid")
sns.barplot(stat_count.values,stat_count.index, alpha=0.9)
plt.title('Frequency Distribution Empty Cells with respect to Features')
plt.xlabel('Number of Miss', fontsize=12)
plt.ylabel('Features', fontsize=12)
plt.show()
```

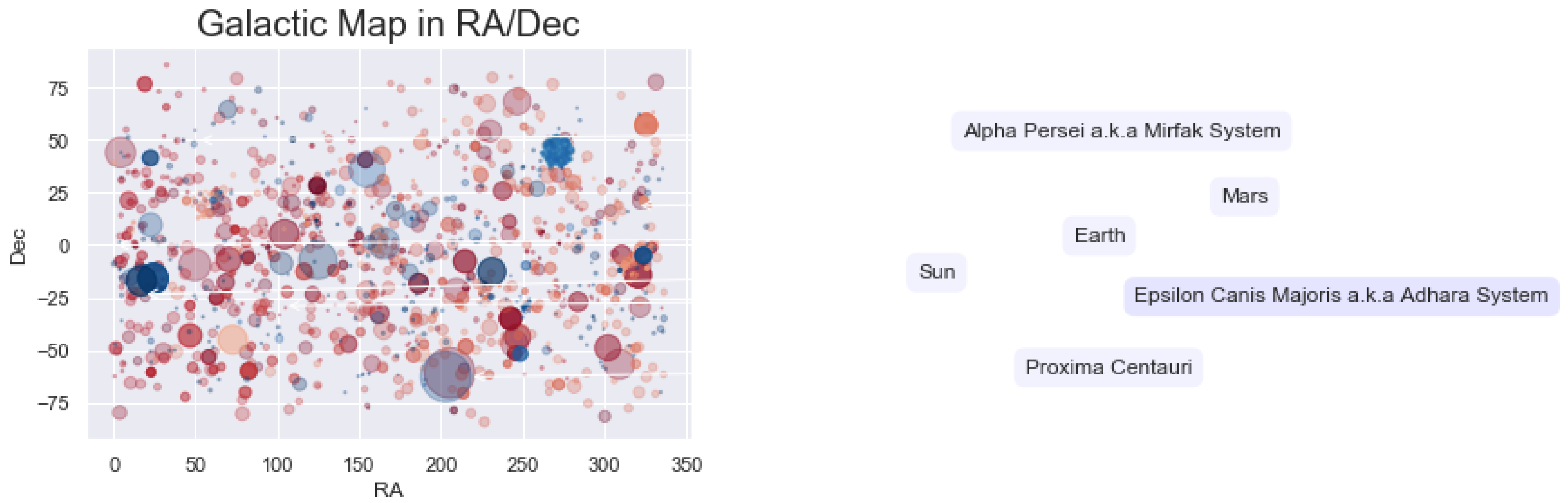


Some visualisations regarding the database

In [8]:
```
x = allExoplanets['S. RA (hrs)']*14
y = allExoplanets['S. DEC (deg)']
#area = np.pi * allExoplanets['S. Radius (SU)']**2
csp=plt.cm.RdYlBu(np.linspace(0,1,len(allExoplanets)))


fig,ax = plt.subplots(figsize=(14,7))
dists = allExoplanets['S. Distance (pc)']
dists.fillna(value=np.mean(dists),inplace=True)
ax=plt.scatter(x,y,s=1000/dists**.5,alpha=0.3,c=csp,cmap=cm.coolwarm)
#ax=plt.scatter(x, y, s=area, c = colors, cmap = colormap, alpha=0.3)
plt.xlabel('RA')
plt.ylabel('Dec')
plt.title('Galactic Map in RA/Dec',size=20)
plt.figure(1,figsize=(16,12))

plt.show()
```



In [9]:
```
## x = allExoplanets['S. RA (hrs)']*14
y = allExoplanets['S. DEC (deg)']
#area = np.pi * allExoplanets['S. Radius (SU)']**2
csp=plt.cm.RdBu(np.linspace(0,1,len(allExoplanets)))


fig,ax = plt.subplots()
dists = allExoplanets['S. Distance (pc)']
dists.fillna(value=np.mean(dists),inplace=True)
ax=plt.scatter(x,y,s=1000/dists,alpha=0.3,c=csp,cmap=cm.coolwarm)
#ax=plt.scatter(x, y, s=area, c = colors, cmap = colormap, alpha=0.3)
plt.xlabel('RA')
plt.ylabel('Dec')
plt.title('Galactic Map in RA/Dec',size=20)
plt.figure(1,figsize=(16,12))




plt.annotate('Earth', xy = (0,0),
             xytext = (560, 0),
             textcoords = 'offset points', ha = 'right', va = 'bottom',
             bbox = dict(boxstyle = 'round,pad=0.5', fc = 'blue', alpha = 0.05),
             arrowprops = dict(arrowstyle = '->',connectionstyle = 'arc3,rad=0')
             )

plt.annotate('Mars', xy = (319.3208,18.6386),
             xytext = (350, 0),
             textcoords = 'offset points', ha = 'right', va = 'bottom',
             bbox = dict(boxstyle = 'round,pad=0.5', fc = 'blue', alpha = 0.05),
             arrowprops = dict(arrowstyle = '->',connectionstyle = 'arc3,rad=0')
             )

plt.annotate('Sun', xy = (18,-23.5),
             xytext = (450, 7),
             textcoords = 'offset points', ha = 'right', va = 'bottom',
             bbox = dict(boxstyle = 'round,pad=0.5', fc = 'blue', alpha = 0.05),
             arrowprops = dict(arrowstyle = '->',connectionstyle = 'arc3,rad=0')
             )

plt.annotate('Proxima Centauri', xy = (217.4292,-62.6794),
             xytext = (400, 0),
             textcoords = 'offset points', ha = 'right', va = 'bottom',
             bbox = dict(boxstyle = 'round,pad=0.5', fc = 'blue', alpha = 0.05),
             arrowprops = dict(arrowstyle = '->',connectionstyle = 'arc3,rad=0')
             )

plt.annotate('Alpha Persei a.k.a Mirfak System', xy = (51.0792,49.8611),
             xytext = (600, 0),
             textcoords = 'offset points', ha = 'right', va = 'bottom',
             bbox = dict(boxstyle = 'round,pad=0.5', fc = 'blue', alpha = 0.05),
             arrowprops = dict(arrowstyle = '->',connectionstyle = 'arc3,rad=0')
             )

plt.annotate('Epsilon Canis Majoris a.k.a Adhara System', xy = (104.6583,-28.9719),
             xytext = (700, 0),
             textcoords = 'offset points', ha = 'right', va = 'bottom',
             bbox = dict(boxstyle = 'round,pad=0.5', fc = 'blue', alpha = 0.1),
             arrowprops = dict(arrowstyle = '->',connectionstyle = 'arc3,rad=0')
             )


plt.show()
```



Visualization

- Size

> Size is denoting the relative distance from earth. actual size is normalized for better visualisation

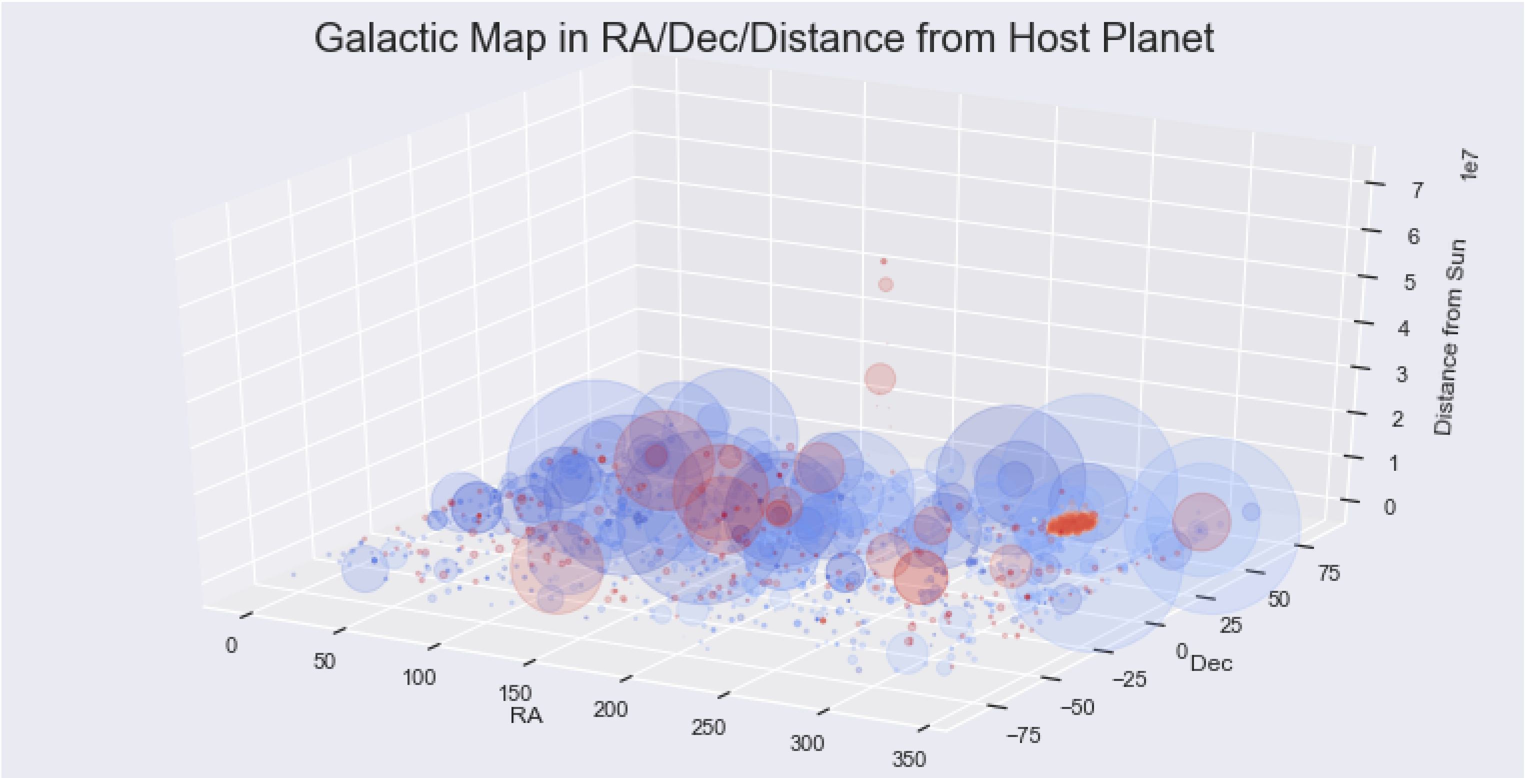
- Color

> color incrementation is used for better visualisation and it increments over all datapoints.

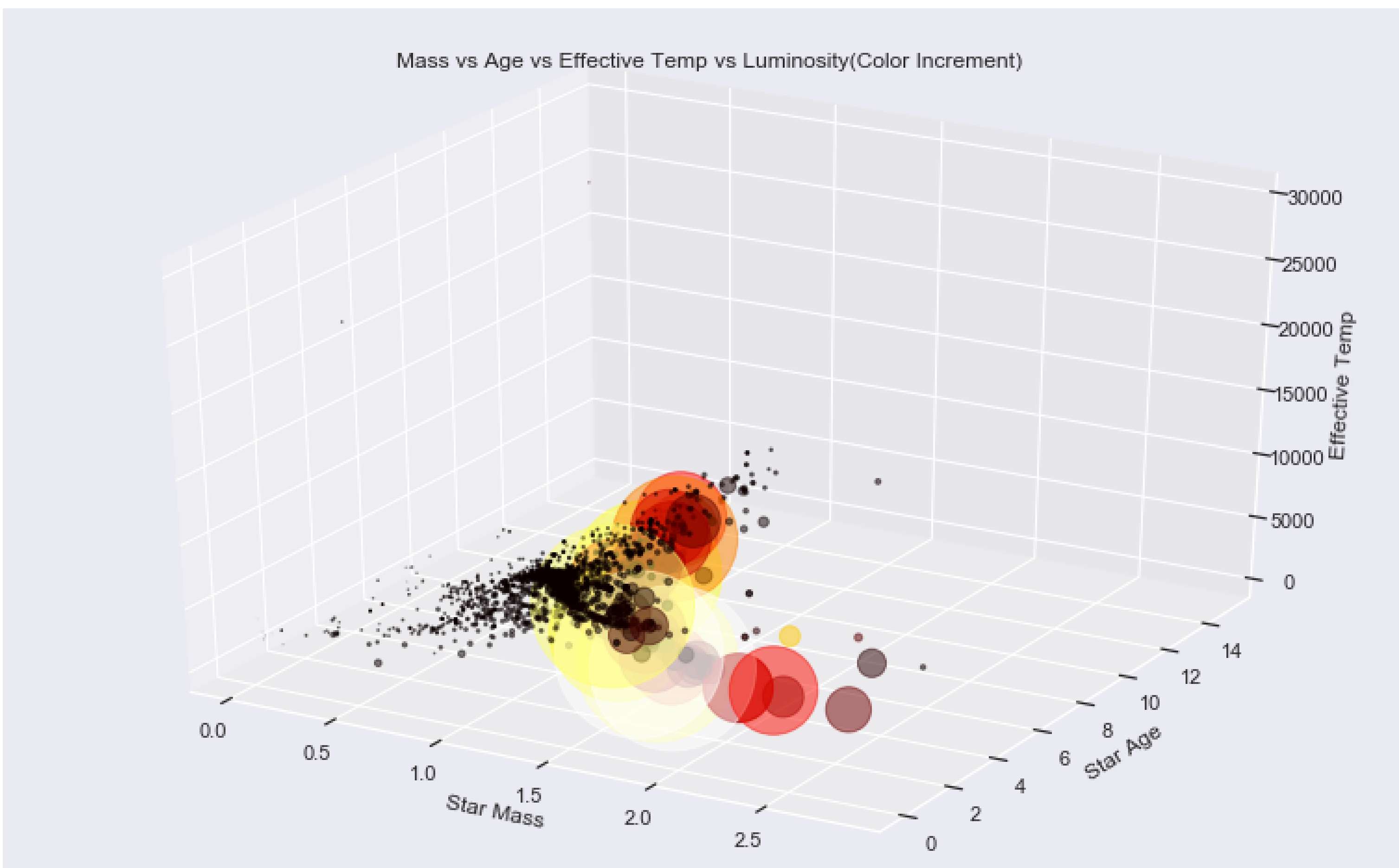
- Filling Up procedure

> missing distance values are filled with mean value of distance

In [10]:
```
x = allExoplanets['S. RA (hrs)']*14
y = allExoplanets['S. DEC (deg)']
z = allExoplanets['S. Distance (pc)']**2
area = np.pi * allExoplanets['S. Radius (SU)']**2
sl = plt.cm.coolwarm(np.linspace(0,1,len(allExoplanets)))
#csp2=plt.cm.(oec['HostStarTempK'])

fig = plt.figure(figsize=(30,50))
ax = fig.add_subplot(621, projection='3d')

ax.axis('on')
ax.scatter(x,y,z,cmap=cm.coolwarm,alpha=0.2,s=area,c=sl)
ax.set_xlabel('RA ')
ax.set_ylabel('Dec ')
ax.set_zlabel('Distance from Sun')
plt.title('Galactic Map in RA/Dec/Distance from Host Planet',size=20)
plt.show()
```



In [11]:
```
x2 = allExoplanets['S. Mass (SU)']
y2 = allExoplanets['S. Age (Gyrs)']
z2 = allExoplanets['S. Teff (K)']

area2 = np.pi * allExoplanets['S. Radius (SU)']**2

#x1 = oec['RA_dd']
#y1 = oec['Dec_dd']
#z1 = dists/1000
#area = np.pi * oec['HostStarRadiusSlrRad']**2

#csp2=plt.cm.(oec['HostStarTempK'])

fig = plt.figure(figsize=(30,40))
ax = fig.add_subplot(421, projection='3d')

ax.axis('on')
ax.scatter(x2,y2,z2,cmap=cm.hot,alpha=0.5,s=area2,c=allExoplanets['S. Luminosity (SU)'])
ax.set_xlabel('Star Mass')
ax.set_ylabel('Star Age')
ax.set_zlabel('Effective Temp')
plt.title('Mass vs Age vs Effective Temp vs Luminosity(Color Increment)')
plt.show()
```



Data Preprocessing

PHL-EC is a very complex and sensitive dataset. Every observation is recorded with high accuracy, That's why we need to perform sensitivity analysis before cleaning, imputing, scaling any part of it.

Detection of Habitable Exoplanets

habstar or habitability, is currently defined as an area, such as a planet or a moon, where liquid water can exist for at least a short duration of time

- A "habitable" planet should:
  - Orbit a star that remains stable in output for billions of years
  - Be at a distance from the star that results in its achieving a suitable temperature so its surface water is liquid, not frozen
  - Have a circular orbit, so constant conditions prevail for its entire "year"
  - Not orbit a star that is too close to a cosmic explosion like a supernova
  - Be far enough from massive planets that they do not continually divert asteroids to hit it or perturb its orbit strongly
  - Probably not be so massive that it retains hydrogen and becomes a "gas giant"
  - Perhaps it is also essential to have a massive planet well outside its orbit, like Jupiter, to divert potential devastating asteroids away, or to make them destroy themselves (as in the asteroid belt).

Habitability

> habstar or habitability, is currently defined as an area, such as a planet or a moon, where liquid water can exist for at least a short duration of time

Solar Twin

> A true solar twins as noted by the Lowell Observatory should have a temperature within ~10 K of the Sun. Space Telescope Science Institute, Lowell Observatory, noted in 1996 that temperature precision of ~10 K can be measured. A temperature of ~10 K reduces the solar twin list to near zero, so ±50 K is used for the chart

Types of Planets

In PHL-EC, Planets are classified into five categories. This classification has been done on the basis of their thermal properties.

Mesoplanets [Asimov - 1989]:

The planetary bodies whose sizes lie between Mercury and Ceres falls under this category (smaller than Mercury and larger than Ceres). These are also referred to as M-planets [Méndez - 2011]. These planets have mean global surface temperature between 0◦C to 50◦C, a necessary condition for complex terrestrial life. These are generally referred as Earth-like planets.

Psychroplanets [Méndez - 2011]:

These planets have mean global surface temperature between -50◦C to 0◦C. Hence, the temperature is colder than optimal for sustenance of terrestrial life

Non-Habitable:

Planets other than mesoplanets and psychroplanets do not have thermal properties required to sustain life.

Description Of Targeted Region

In [12]:
```
print(allExoplanets['P. Habitable Class'].describe())
```

```
count              3875
unique                5
top       non-habitable
freq               3820
Name: P. Habitable Class, dtype: object
```

In [13]:
```
print('Our Dataset,  or Target region if I be more precise, has been classified into 5 categories, which ar
e:')
for i in allExoplanets['P. Habitable Class'].unique():
    print(i)
```

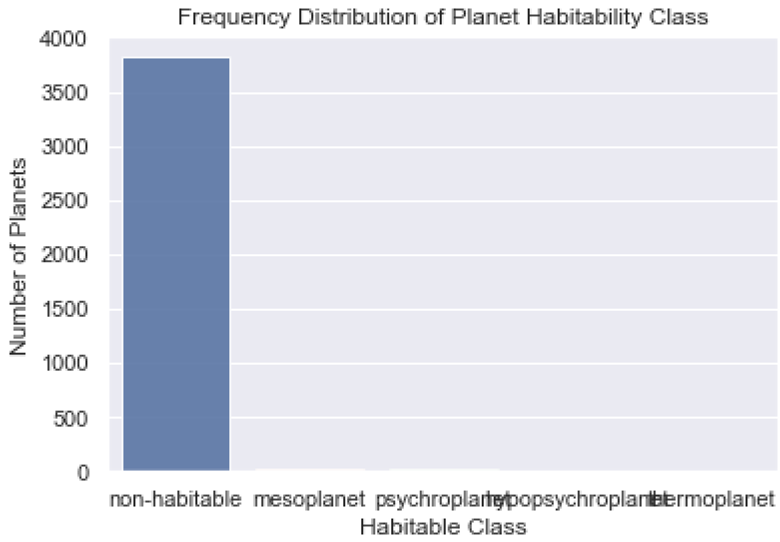
```
Our Dataset,  or Target region if I be more precise, has been classified into 5 categories, which are:
non-habitable
psychroplanet
mesoplanet
thermoplanet
hypopsychroplanet
```

In [14]:
```
print(allExoplanets['P. Habitable Class'].value_counts())
```

```
non-habitable        3820
mesoplanet             31
psychroplanet          18
hypopsychroplanet       3
thermoplanet            3
Name: P. Habitable Class, dtype: int64
```

Last three output clearly states that 'non - habitable' class is dominating over other classes. We should visualise the frequency to understand the issue more deeply.

In [15]:
```
stat_count = allExoplanets['P. Habitable Class'].value_counts()
sns.set(style="darkgrid")
sns.barplot(stat_count.index, stat_count.values, alpha=0.9)
plt.title('Frequency Distribution of Planet Habitability Class')
plt.ylabel('Number of Planets', fontsize=12)
plt.xlabel('Habitable Class', fontsize=12)
plt.show()
```



Thermoplanet:

> A class of planets, which has a temperature in the range of 50◦C-100◦C. This is warmer than the temperature range suited for most terrestrial life [Méndez2011].

Hypopsychroplanets:

> A class of planets whose temperature is below −50◦C. Planets belonging to this category are too cold for the survival of most terrestrial life [Méndez2011].

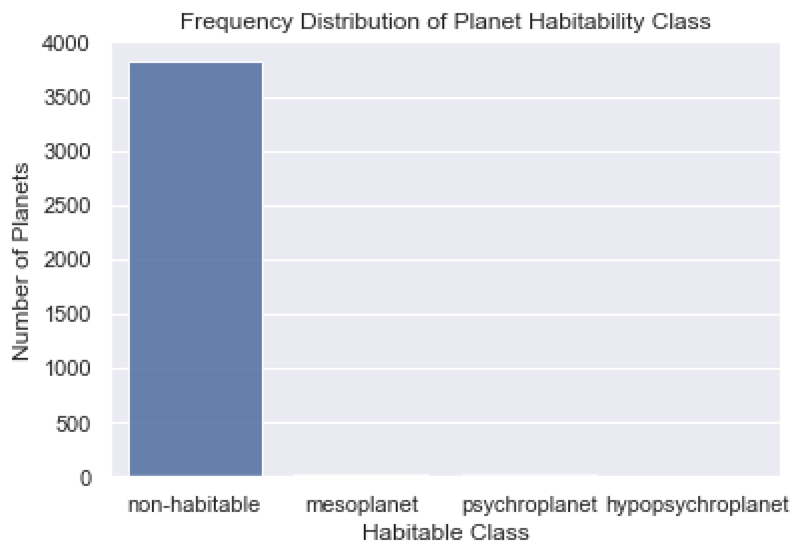
NOTE

```
The above two classes have three data entities each in the augmented data set used. This
number is inadequate for the task of classification, and hence the total of six entities
were excluded from the experiment.
```

In [16]:
```
allExoplanets=allExoplanets[allExoplanets.iloc[:,7]!= 'thermoplanet']
allExoplanets['P. Habitable Class'].value_counts()
```

Out[16]:
```
non-habitable        3820
mesoplanet             31
psychroplanet          18
hypopsychroplanet       3
Name: P. Habitable Class, dtype: int64
```

In [17]:
```
stat_count = allExoplanets['P. Habitable Class'].value_counts()
sns.set(style="darkgrid")
sns.barplot(stat_count.index, stat_count.values, alpha=0.9)
plt.title('Frequency Distribution of Planet Habitability Class')
plt.ylabel('Number of Planets', fontsize=12)
plt.xlabel('Habitable Class', fontsize=12)
plt.show()
```



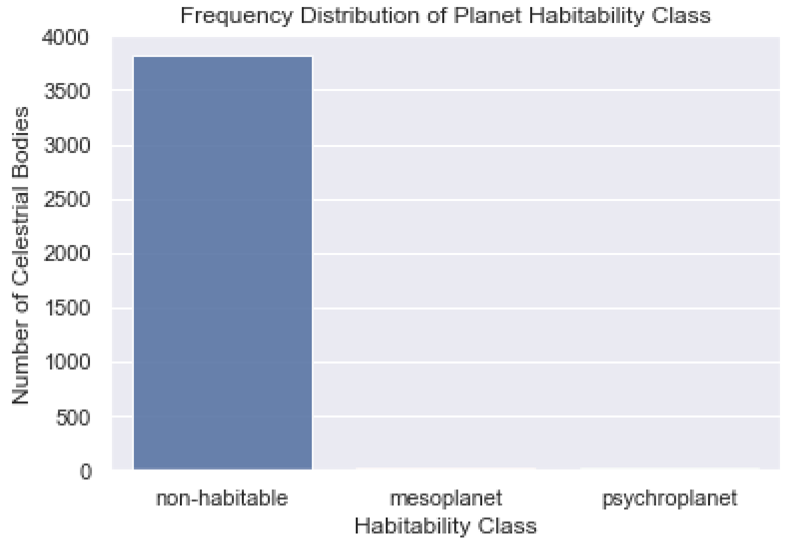
In [18]:
```
allExoplanets = allExoplanets[allExoplanets.iloc[:,7]!= 'hypopsychroplanet']
allExoplanets['P. Habitable Class'].value_counts()
```

Out[18]:
```
non-habitable    3820
mesoplanet         31
psychroplanet      18
Name: P. Habitable Class, dtype: int64
```

In [19]:
```
allExoplanets['P. Habitable Class'].value_counts()
```

Out[19]:
```
non-habitable    3820
mesoplanet         31
psychroplanet      18
Name: P. Habitable Class, dtype: int64
```

In [20]:
```
stat_count = allExoplanets['P. Habitable Class'].value_counts()
sns.set(style="darkgrid")
sns.barplot(stat_count.index, stat_count.values, alpha=0.9)
plt.title('Frequency Distribution of Planet Habitability Class')
plt.ylabel('Number of Celestrial Bodies', fontsize=12)
plt.xlabel('Habitability Class', fontsize=12)
plt.show()
```



thermoplanet and hypopsychroplanet instances are successfully removed

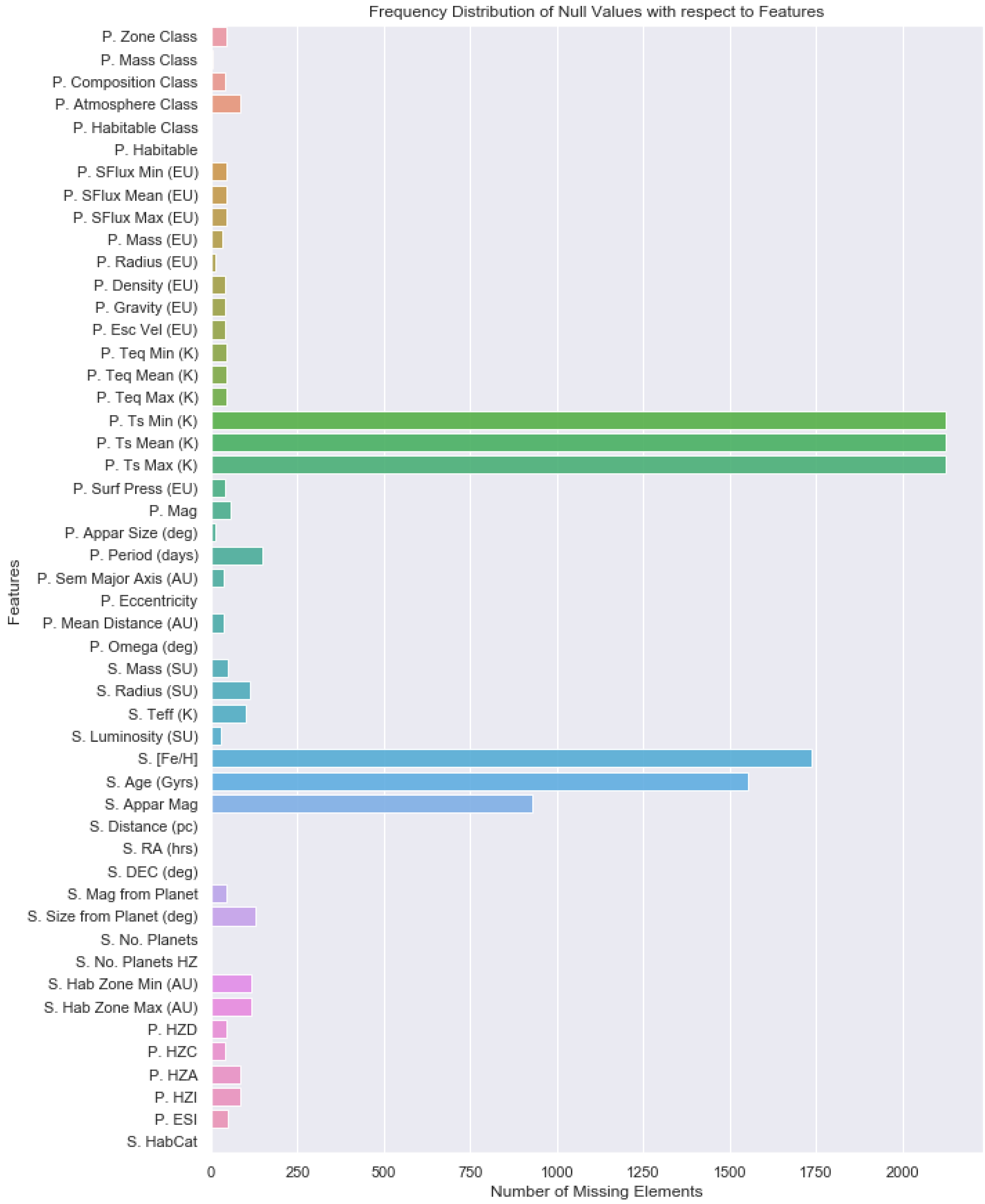
Note

Although thermoplanet and hypopsychroplanet class have been removed from the set for being the most submissive ones. Still, the ratio hasn't improved very much. This may result in High Accuracy and False Positives.

Feature Selection

P.NameKepler (planet's name), Sname HD and Sname Hid (name of parent star), S.constellation (name of constellation), Stype (type of parent star), P.SPH (planet standard primary habitability), P.interior ESI (interior earth similarity index), P.surface ESI (surface earth similarity index), P.disc method (method of discovery of planet), P.disc year (year of discovery of planet), P. Max Mass, P. Min Mass, P.inclination and P.Hab Moon (flag indicating planet's potential as a habitable exomoons) were removed as these attributes do not contribute to the nature of classification of habitability of a planet. Interior ESI and surface ESI, however, together contribute to habitability, but since the data set directly provides P.ESI, these two features were neglected. Following this, classification algorithms were applied on the processed data set. In all, 50 features are used.

In [21]:
```
pred = ['P. Zone Class','P. Mass Class','P. Composition Class','P. Atmosphere Class',
                'P. Habitable Class','P. Habitable',


                'P. SFlux Min (EU)','P. SFlux Mean (EU)','P. SFlux Max (EU)',
                'P. Mass (EU)','P. Radius (EU)','P. Density (EU)','P. Gravity (EU)',
                'P. Esc Vel (EU)','P. Teq Min (K)','P. Teq Mean (K)','P. Teq Max (K)',
                'P. Ts Min (K)','P. Ts Mean (K)','P. Ts Max (K)','P. Surf Press (EU)',
                'P. Mag','P. Appar Size (deg)','P. Period (days)','P. Sem Major Axis (AU)',
                'P. Eccentricity','P. Mean Distance (AU)','P. Omega (deg)','S. Mass (SU)',
                'S. Radius (SU)','S. Teff (K)','S. Luminosity (SU)','S. [Fe/H]','S. Age (Gyrs)',
                'S. Appar Mag','S. Distance (pc)','S. RA (hrs)','S. DEC (deg)',
                'S. Mag from Planet','S. Size from Planet (deg)','S. No. Planets',
                'S. No. Planets HZ','S. Hab Zone Min (AU)','S. Hab Zone Max (AU)',
                'P. HZD','P. HZC','P. HZA','P. HZI','P. ESI','S. HabCat']

pred_cat = ['P. Zone Class','P. Mass Class','P. Composition Class','P. Atmosphere Class',
                'P. Habitable Class','P. Habitable']
pred_num = ['P. SFlux Min (EU)','P. SFlux Mean (EU)','P. SFlux Max (EU)','P. Mass (EU)',
            'P. Radius (EU)','P. Density (EU)','P. Gravity (EU)',
                'P. Esc Vel (EU)','P. Teq Min (K)','P. Teq Mean (K)','P. Teq Max (K)',
                'P. Ts Min (K)','P. Ts Mean (K)','P. Ts Max (K)','P. Surf Press (EU)',
                'P. Mag','P. Appar Size (deg)','P. Period (days)','P. Sem Major Axis (AU)',
                'P. Eccentricity','P. Mean Distance (AU)','P. Omega (deg)','S. Mass (SU)',
                'S. Radius (SU)','S. Teff (K)','S. Luminosity (SU)','S. [Fe/H]','S. Age (Gyrs)',
                'S. Appar Mag','S. Distance (pc)','S. RA (hrs)','S. DEC (deg)',
                'S. Mag from Planet','S. Size from Planet (deg)','S. No. Planets',
                'S. No. Planets HZ','S. Hab Zone Min (AU)','S. Hab Zone Max (AU)',
                'P. HZD','P. HZC','P. HZA','P. HZI','P. ESI','S. HabCat']

examine = pd.DataFrame()
examine =  allExoplanets[pred]


f, ax = plt.subplots(figsize=(10, 15))

stat_count = examine.isnull().sum()
sns.set(style="darkgrid")
sns.barplot(stat_count.values,stat_count.index, alpha=0.9)
plt.title('Frequency Distribution of Null Values with respect to Features')
plt.xlabel('Number of Missing Elements', fontsize=12)
plt.ylabel('Features', fontsize=12)
plt.show()
```


Frequency Distribution of Null Values with respect to Features
P. Zone Class
P. Mass Class
P. Composition Class
P. Atmosphere Class
P. Habitable Class
P. Habitable
P. SFlux Min (EU)
P. SFlux Mean (EU)
P. SFlux Max (EU)
P. Mass (EU)
P. Radius (EU)
P. Density (EU)
P. Gravity (EU)
P. Esc Vel (EU)
P. Teq Min (K)
P. Teq Mean (K)
P. Teq Max (K)
P. Ts Min (K)
P. Ts Mean (K)
P. Ts Max (K)
P. Surf Press (EU)
P. Mag
P. Appar Size (deg)
P. Period (days)
P. Sem Major Axis (AU)
P. Eccentricity
P. Mean Distance (AU)
P. Omega (deg)
S. Mass (SU)
S. Radius (SU)
S. Teff (K)
S. Luminosity (SU)
S. [Fe/H]
S. Age (Gyrs)
S. Appar Mag
S. Distance (pc)
S. RA (hrs)
S. DEC (deg)
S. Mag from Planet
S. Size from Planet (deg)
S. No. Planets
S. No. Planets HZ
S. Hab Zone Min (AU)
S. Hab Zone Max (AU)
P. HZD
P. HZC
P. HZA
P. HZI
P. ESI
S. HabCat
Features
0
250
500
750
1000
1250
1500
1750
2000
Number of Missing Elements

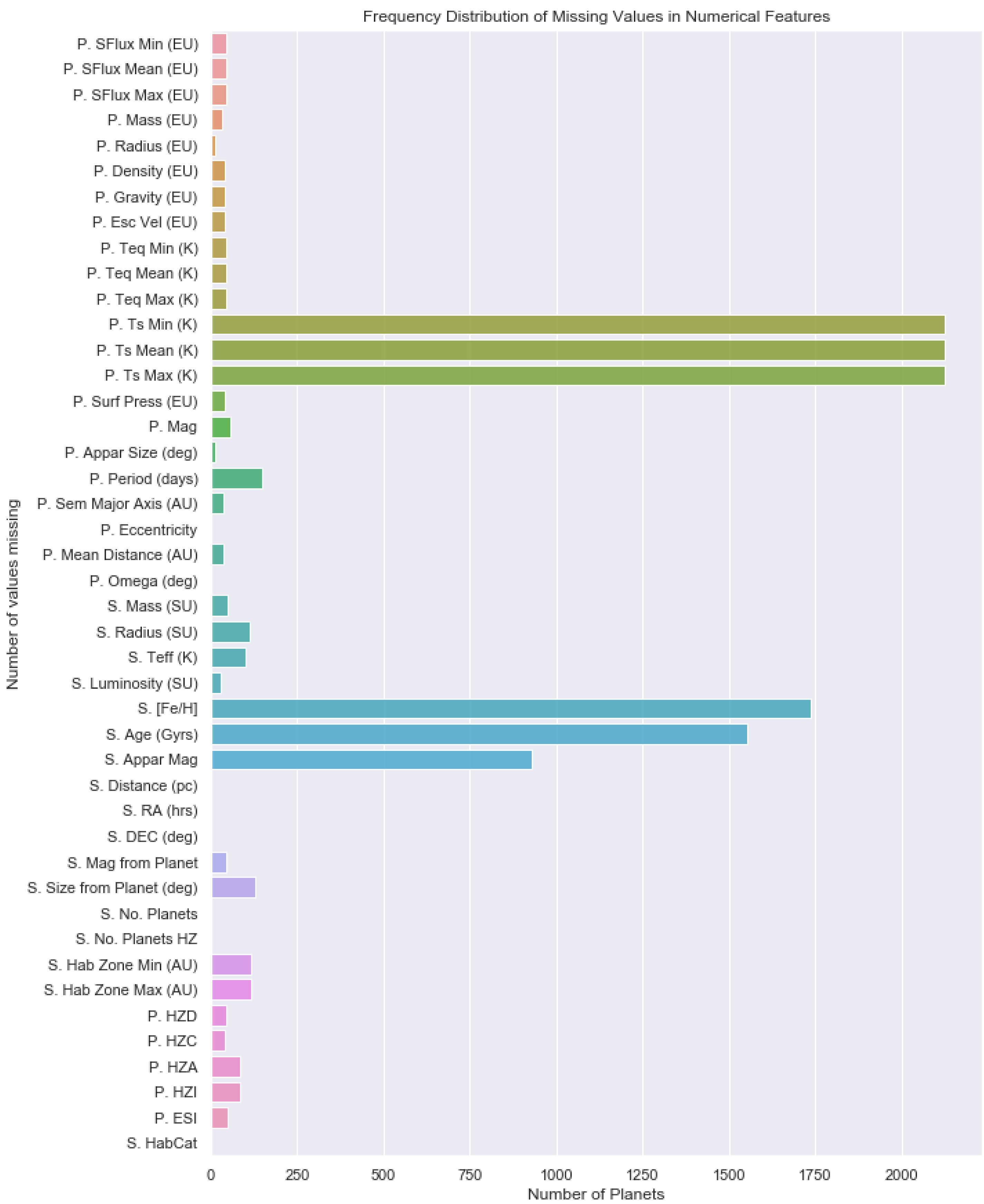
In [22]:
```
cat1 = len(examine.select_dtypes(include=['object']).columns)
num1 = len(examine.select_dtypes(include=['int64','float64']).columns)
print('Features of examine consists of ', cat, 'categorical', ' and ',
      num, 'numerical features')
print('\n\nCategorical Features:\n ')
for i in pred_cat:
    print("{feature}".format(feature=i),end='\t')
print('\n\nNumerical Features:\n ')
for i in pred_num:
    print("{feature}".format(feature=i))
```

```
Features of examine consists of  14 categorical  and  54 numerical features


Categorical Features:

P. Zone Class   P. Mass Class   P. Composition Class    P. Atmosphere Class     P. Habitable Class      P. Ha
bitable

Numerical Features:

P. SFlux Min (EU)
P. SFlux Mean (EU)
P. SFlux Max (EU)
P. Mass (EU)
P. Radius (EU)
P. Density (EU)
P. Gravity (EU)
P. Esc Vel (EU)
P. Teq Min (K)
P. Teq Mean (K)
P. Teq Max (K)
P. Ts Min (K)
P. Ts Mean (K)
P. Ts Max (K)
P. Surf Press (EU)
P. Mag
P. Appar Size (deg)
P. Period (days)
P. Sem Major Axis (AU)
P. Eccentricity
P. Mean Distance (AU)
P. Omega (deg)
S. Mass (SU)
S. Radius (SU)
S. Teff (K)
S. Luminosity (SU)
S. [Fe/H]
S. Age (Gyrs)
S. Appar Mag
S. Distance (pc)
S. RA (hrs)
S. DEC (deg)
S. Mag from Planet
S. Size from Planet (deg)
S. No. Planets
S. No. Planets HZ
S. Hab Zone Min (AU)
S. Hab Zone Max (AU)
P. HZD
P. HZC
P. HZA
P. HZI
P. ESI
S. HabCat
```

cat1 = len(examine.select_dtypes(include=['object']).columns) num1 = len(examine.select_dtypes(include=['int64','float64']).columns) print('Features of examine consists of ', cat, 'categorical', ' and ', num, 'numerical features') print('\n\nCategorical Features:\n ') for i in pred_cat: print(" {feature}".format(feature=i),end='\t') print('\n\nNumerical Features:\n ') for i in pred_num: print("{feature}".format(feature=i),end='\t')

In [23]:
```
print("Total of null Values in each Feature of Numerical Features region: \n")
#imputed_exnum=examine.iloc[:,8:]
#print(examine.iloc[:,8:].isnull().sum())
fig , ax = plt.subplots(figsize=(10,15))
stat_count = examine.iloc[:,6:].isnull().sum()
sns.set(style="darkgrid")
sns.barplot(stat_count.values, stat_count.index, alpha=0.9)
plt.title('Frequency Distribution of Missing Values in Numerical Features')
plt.xlabel('Number of Planets', fontsize=12)
plt.ylabel('Number of values missing', fontsize=12)
plt.show()
```

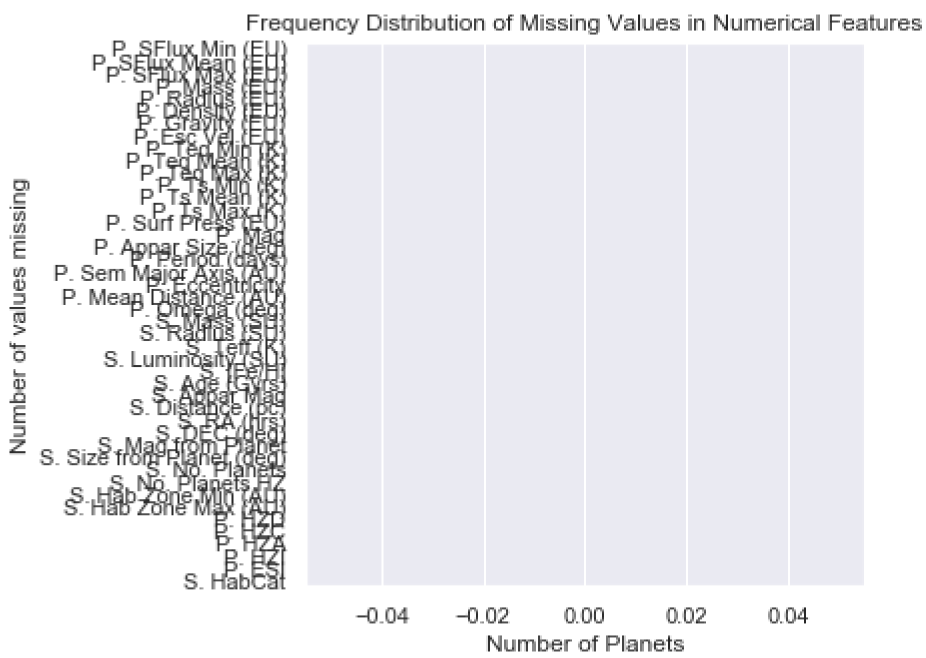
```
Total of null Values in each Feature of Numerical Features region:
```



In [24]:
```
from sklearn.preprocessing import Imputer
mean_imputer = Imputer(missing_values='NaN', strategy='mean', axis=0)

# Training imputer on the numerical region of 'Examine'
mean_imputer = mean_imputer.fit(examine.iloc[:,6:])
examine.iloc[:,6:] = mean_imputer.transform(examine.iloc[:,6:])

numpart = examine.iloc[:,6:]
```

```
C:\Users\User\Anaconda3\lib\site-packages\pandas\core\indexing.py:543: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view
-versus-copy
  self.obj[item] = s
```

In [25]:
```
#print(examine.iloc[:,8:].isnull().sum())

fig , ax = plt.subplots(figsize=(5,5))
stat_count = examine.iloc[:,6:].isnull().sum()
sns.set(style="darkgrid")
sns.barplot(stat_count.values, stat_count.index, alpha=0.9)
plt.title('Frequency Distribution of Missing Values in Numerical Features')
plt.xlabel('Number of Planets', fontsize=12)
plt.ylabel('Number of values missing', fontsize=12)
plt.show()
```

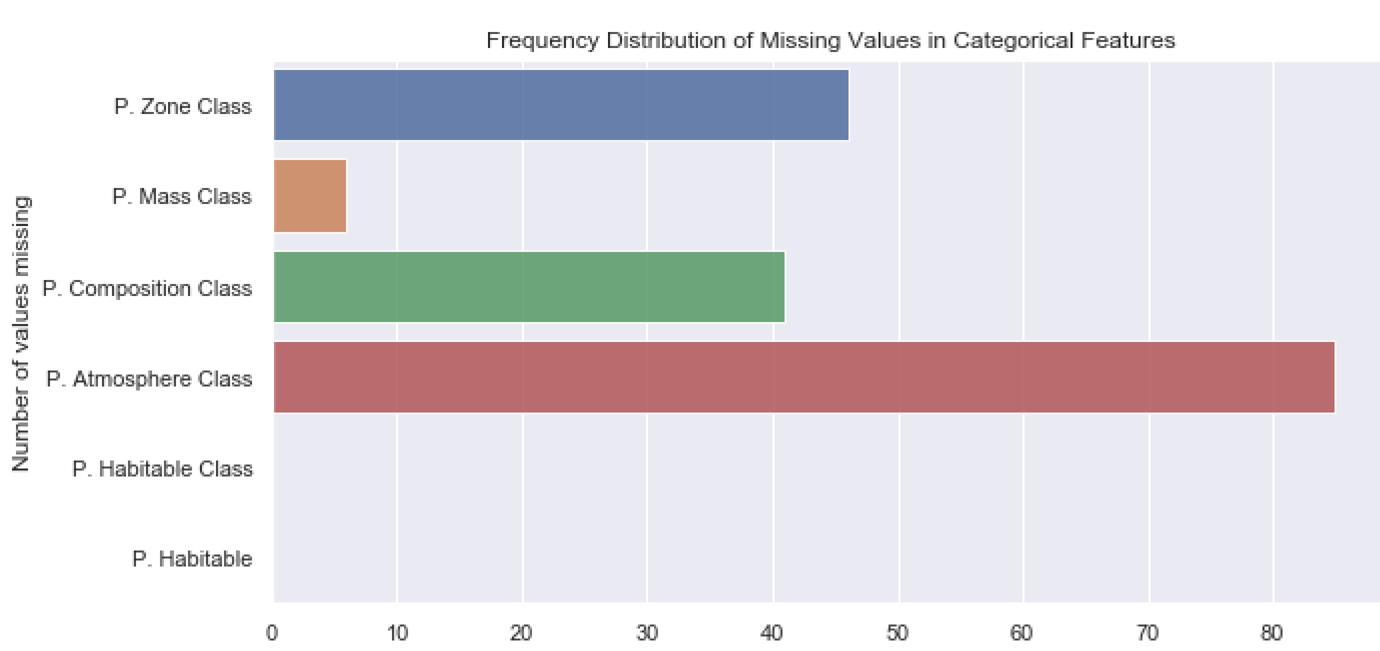


Imputing of Categorical Features

Categorical Features are imputed by most frequent strategy

In [26]:
```
#imputed_exnum=examine.iloc[:,8:]
#print(examine.iloc[:,:8].isnull().sum())
#f, ax = plt.subplots(figsize=(5, 5))

###for i in examine[pred_cat]:
###    stat_count = examine[i].value_counts()
###    sns.set(style="darkgrid")
###    sns.barplot(stat_count.index, stat_count.values, alpha=0.9)
###    plt.title('Frequency Distribution of {col}'.format(col=i))
###    plt.ylabel('Number of Planets', fontsize=12)
###    plt.xlabel('{s}'.format(s=i), fontsize=12)
###plt.show()
print("Total of null Values in each Feature of Categorical Features region: \n")
#imputed_exnum=examine.iloc[:,8:]
#print(examine.iloc[:,8:].isnull().sum())
fig , ax = plt.subplots(figsize=(10,5))
stat_count = examine.iloc[:,:6].isnull().sum()
sns.set(style="darkgrid")
sns.barplot(stat_count.values, stat_count.index, alpha=0.9)
plt.title('Frequency Distribution of Missing Values in Categorical Features')
plt.xlabel('Number of Planets', fontsize=12)
plt.ylabel('Number of values missing', fontsize=12)
plt.show()
```

```
Total of null Values in each Feature of Categorical Features region:
```

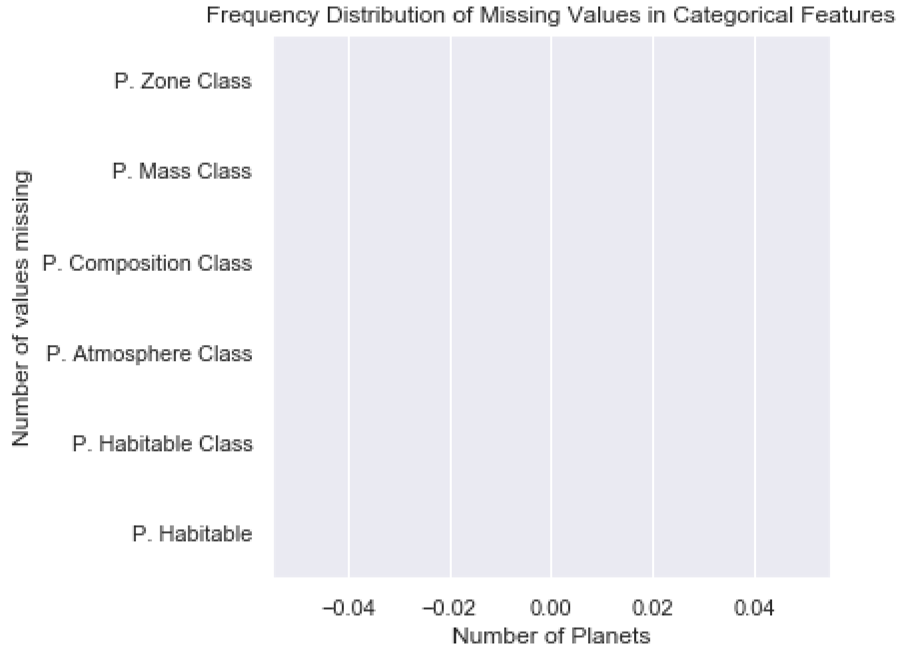


In [27]:
```
#dataframe imputer
from sklearn.base import TransformerMixin

class DataFrameImputer(TransformerMixin):
    def __init__(self):
            #cat -> most frq , num -> mean
        return None
    def fit(self, X, y=None):

        self.fill = pd.Series([X[c].value_counts().index[0]
                               if X[c].dtype == np.dtype('O')
                               else X[c].mean() for c in X],
                                 index = X.columns)

        return self

    def transform(self,X,y=None):
        return X.fillna(self.fill)

examine.iloc[:,:4] = DataFrameImputer().fit_transform(examine.iloc[:,:4])
```

```
C:\Users\User\Anaconda3\lib\site-packages\pandas\core\indexing.py:543: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view
-versus-copy
  self.obj[item] = s
```

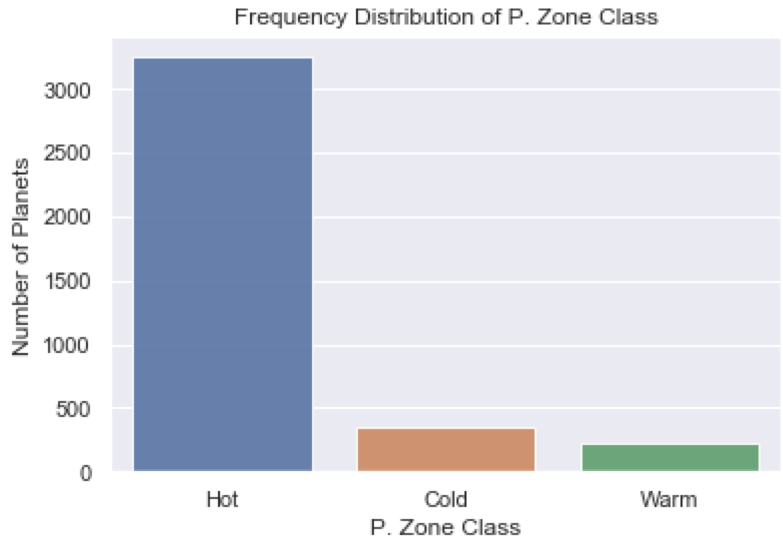
In [28]:
```
catpart = examine.iloc[:,:6]
```

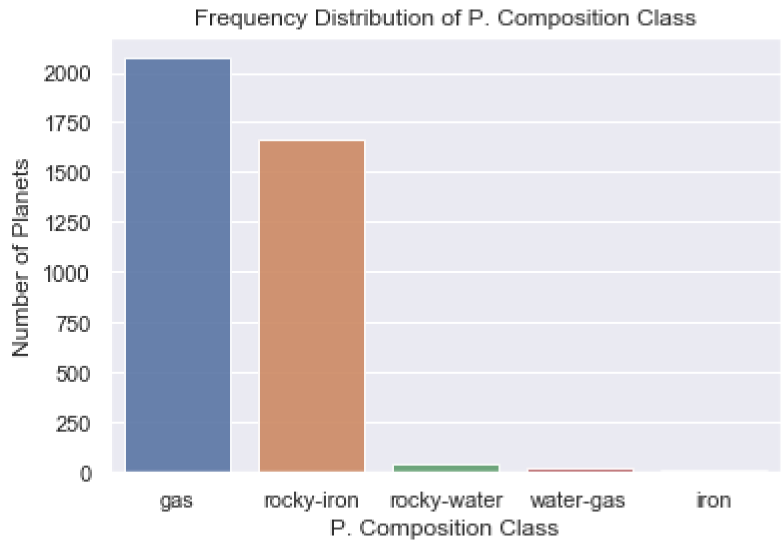
In [29]:
```
#imputed_exnum=examine.iloc[:,8:]
#print(examine.iloc[:,:8].isnull().sum())

fig , ax = plt.subplots(figsize=(5,5))
stat_count = examine.iloc[:,:6].isnull().sum()
sns.set(style="darkgrid")
sns.barplot(stat_count.values, stat_count.index, alpha=0.9)
plt.title('Frequency Distribution of Missing Values in Categorical Features')
plt.xlabel('Number of Planets', fontsize=12)
plt.ylabel('Number of values missing', fontsize=12)
plt.show()
```

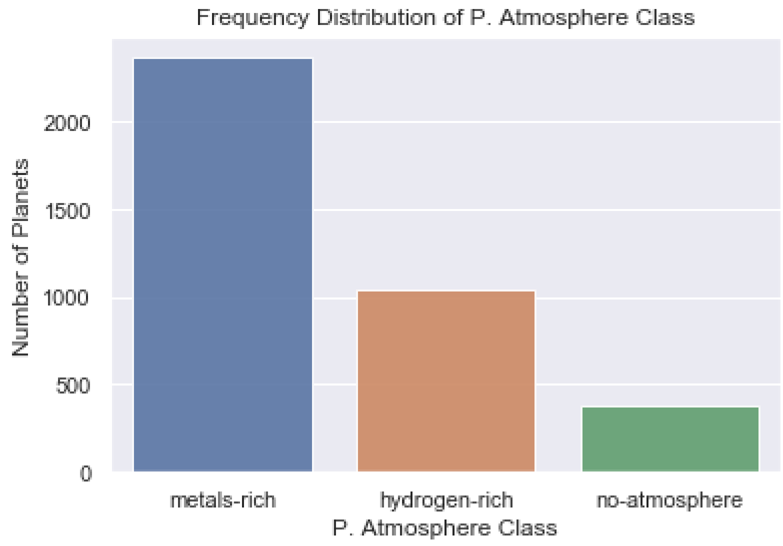


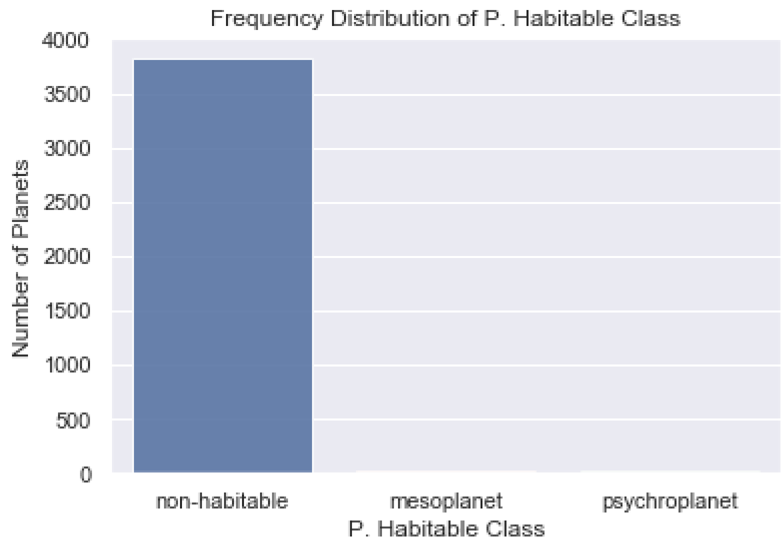
Label Frequency in Categorical Features

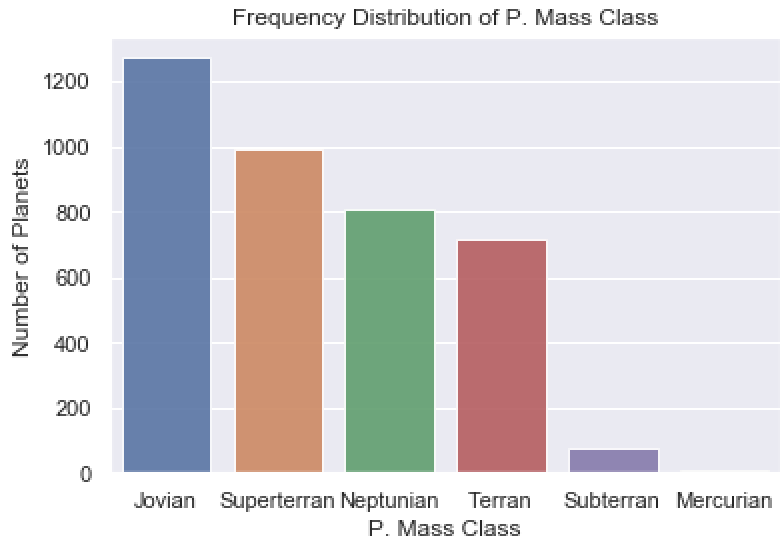
In [30]:
```
stat_count = allExoplanets['P. Zone Class'].value_counts()
sns.set(style="darkgrid")
sns.barplot(stat_count.index, stat_count.values, alpha=0.9)
plt.title('Frequency Distribution of P. Zone Class')
plt.ylabel('Number of Planets', fontsize=12)
plt.xlabel('P. Zone Class', fontsize=12)
plt.show()

stat_count = allExoplanets['P. Composition Class'].value_counts()
sns.set(style="darkgrid")
sns.barplot(stat_count.index, stat_count.values, alpha=0.9)
plt.title('Frequency Distribution of P. Composition Class')
plt.ylabel('Number of Planets', fontsize=12)
plt.xlabel('P. Composition Class', fontsize=12)
plt.show()

stat_count = allExoplanets['P. Atmosphere Class'].value_counts()
sns.set(style="darkgrid")
sns.barplot(stat_count.index, stat_count.values, alpha=0.9)
plt.title('Frequency Distribution of P. Atmosphere Class')
plt.ylabel('Number of Planets', fontsize=12)
plt.xlabel('P. Atmosphere Class', fontsize=12)
plt.show()

stat_count = allExoplanets['P. Habitable Class'].value_counts()
sns.set(style="darkgrid")
sns.barplot(stat_count.index, stat_count.values, alpha=0.9)
plt.title('Frequency Distribution of P. Habitable Class')
plt.ylabel('Number of Planets', fontsize=12)
plt.xlabel('P. Habitable Class', fontsize=12)
plt.show()

stat_count = allExoplanets['P. Mass Class'].value_counts()
sns.set(style="darkgrid")
sns.barplot(stat_count.index, stat_count.values, alpha=0.9)
plt.title('Frequency Distribution of P. Mass Class')
plt.ylabel('Number of Planets', fontsize=12)
plt.xlabel('P. Mass Class', fontsize=12)
plt.show()

stat_count = allExoplanets['P. Habitable'].value_counts()
sns.set(style="darkgrid")
sns.barplot(stat_count.index, stat_count.values, alpha=0.9)
plt.title('Frequency Distribution of Habitability')
plt.ylabel('Number of Planets', fontsize=12)
plt.xlabel('P. Habitable', fontsize=12)
plt.show()
```

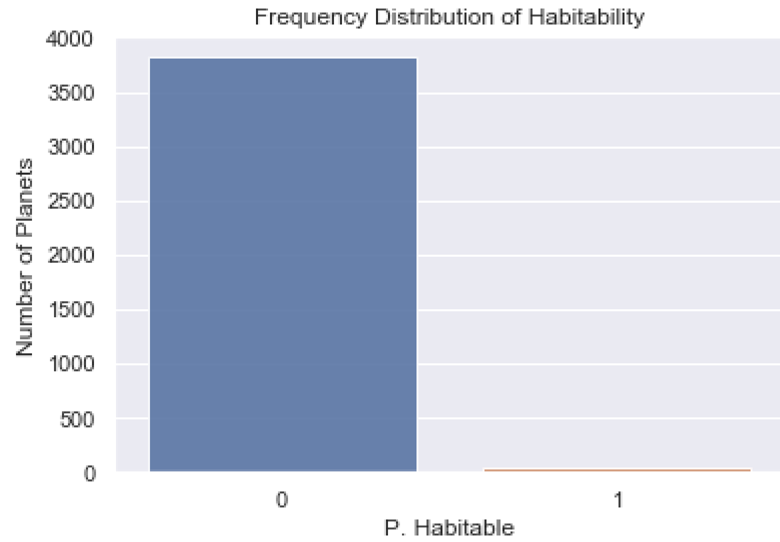

Frequency Distribution of P. Zone Class
Number of Planets
3000
2500
2000
1500
1000
500
0
Hot
Cold
Warm
P. Zone Class


Frequency Distribution of P. Composition Class
Number of Planets
2000
1750
1500
1250
1000
750
500
250
0
gas
rocky-iron
rocky-water
water-gas
iron
P. Composition Class


Frequency Distribution of P. Atmosphere Class
Number of Planets
2000
1500
1000
500
0
metals-rich
hydrogen-rich
no-atmosphere
P. Atmosphere Class


Frequency Distribution of P. Habitable Class
Number of Planets
4000
3500
3000
2500
2000
1500
1000
500
0
non-habitable
mesoplanet
psychroplanet
P. Habitable Class


Frequency Distribution of P. Mass Class
Number of Planets
1200
1000
800
600
400
200
0
Jovian
Superterran
Neptunian
Terran
Subterran
Mercurian
P. Mass Class



Present Status of my database is: #examine2 = DataFrameImputer().fit_transform(examine) ### taking a backup #examine2.to_csv('exoplanets_imputed_notencoded.csv') #examine2 #examine.isna().sum()

All the Features have been imputed. Now, What I'm gonna do is to encode all the variables. Before that, separating the Independent variable, i.e, separating X and y.

Selection of X and y

In [31]:
```
y=examine['P. Habitable Class']
dependent = ['P. Zone Class','P. Mass Class','P. Composition Class','P. Atmosphere Class',
                'P. Habitable',

             'P. SFlux Min (EU)','P. SFlux Mean (EU)','P. SFlux Max (EU)',
                'P. Mass (EU)','P. Radius (EU)','P. Density (EU)','P. Gravity (EU)',
                'P. Esc Vel (EU)','P. Teq Min (K)','P. Teq Mean (K)','P. Teq Max (K)',
                'P. Ts Min (K)','P. Ts Mean (K)','P. Ts Max (K)','P. Surf Press (EU)',
                'P. Mag','P. Appar Size (deg)','P. Period (days)','P. Sem Major Axis (AU)',
                'P. Eccentricity','P. Mean Distance (AU)','P. Omega (deg)','S. Mass (SU)',
                'S. Radius (SU)','S. Teff (K)','S. Luminosity (SU)','S. [Fe/H]','S. Age (Gyrs)',
                'S. Appar Mag','S. Distance (pc)','S. RA (hrs)','S. DEC (deg)',
                'S. Mag from Planet','S. Size from Planet (deg)','S. No. Planets',
                'S. No. Planets HZ','S. Hab Zone Min (AU)','S. Hab Zone Max (AU)',
                'P. HZD','P. HZC','P. HZA','P. HZI','P. ESI','S. HabCat']
X =  examine[dependent]
```

In [32]:
```
print('X and y are of shape '+str(X.shape)+' and '+str(y.shape)+' respectively')
```

```
X and y are of shape (3869, 49) and (3869,) respectively
```

Current Status of our dataset

In [33]:
```
examine.head()
```

Out[33]:

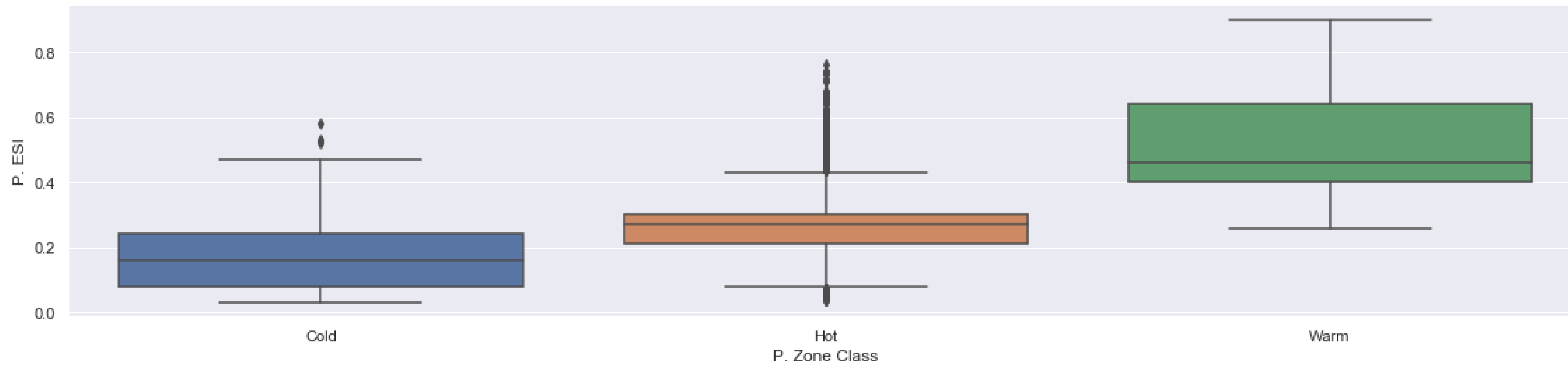
	P. Zone Class	P. Mass Class	P. Composition Class	P. Atmosphere Class	P. Habitable Class	P. Habitable	P. SFlux Min (EU)	P. SFlux Mean (EU)	P. SFlux Max (EU)	P. Mass (EU)	...	S. No. Planets	S Pl
0	Cold	Jovian	gas	hydrogen-rich	non-habitable	0	4.080000e-06	4.080000e-06	4.080000e-06	4451.16	...	1.0	0.(
1	Cold	Jovian	gas	hydrogen-rich	non-habitable	0	2.166914e-02	2.166914e-02	2.166914e-02	6358.80	...	1.0	0.(
2	Cold	Jovian	gas	hydrogen-rich	non-habitable	0	3.960000e-06	3.960000e-06	3.960000e-06	4133.22	...	1.0	0.(
3	Cold	Jovian	gas	hydrogen-rich	non-habitable	0	1.030000e-05	1.030000e-05	1.030000e-05	6358.80	...	1.0	0.(
4	Cold	Jovian	gas	hydrogen-rich	non-habitable	0	2.370000e-07	2.370000e-07	2.370000e-07	4419.37	...	1.0	0.(

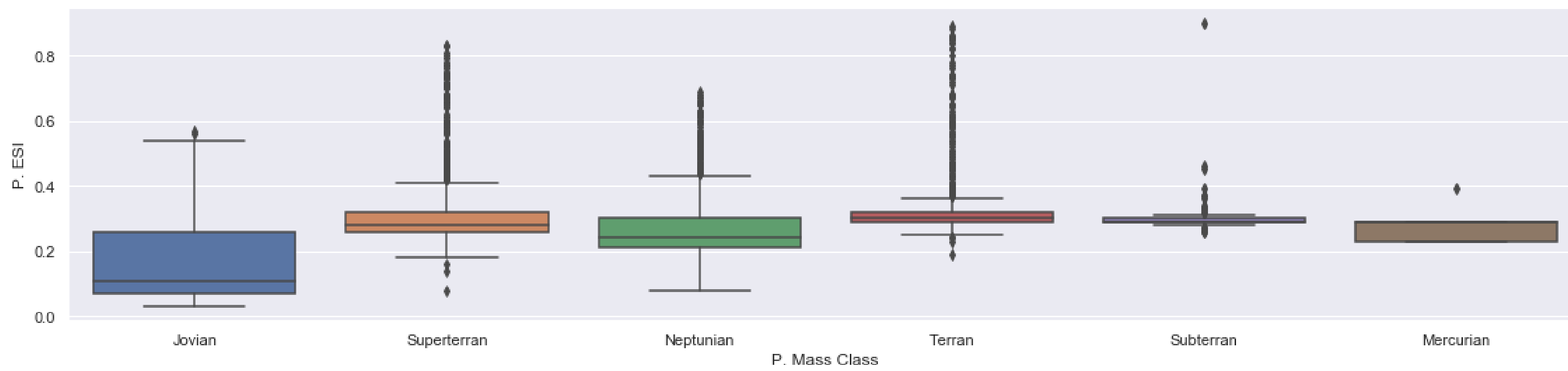
5 rows × 50 columns

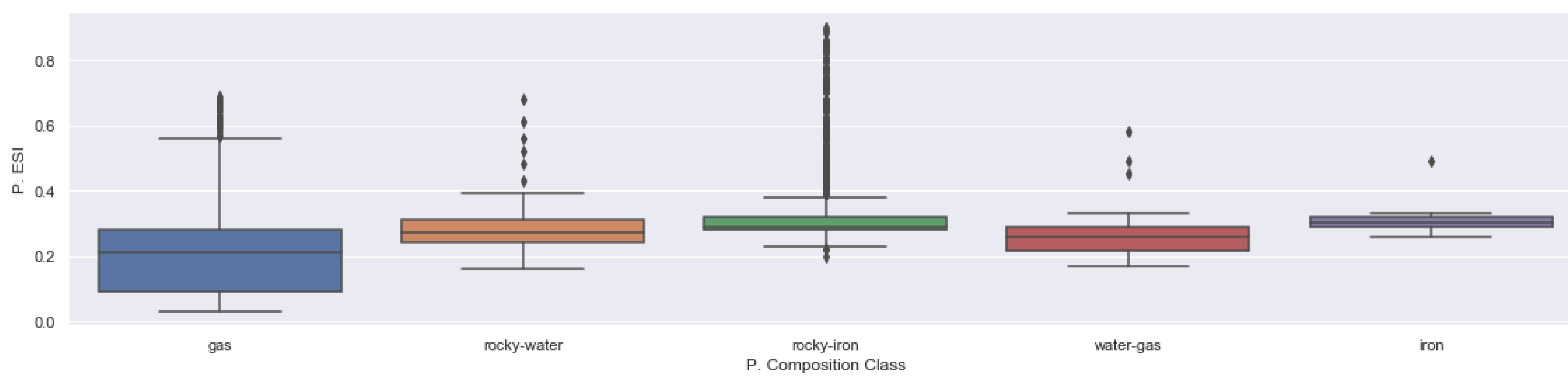
How ESI is correlated with different features

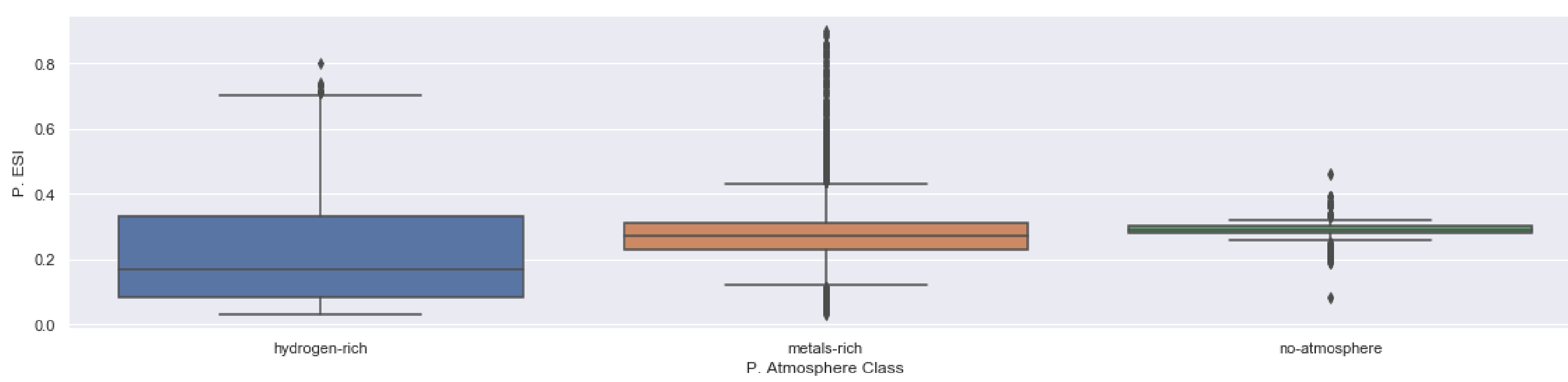
In [34]:
```
plt.figure(figsize=(20,5))

for i in pred_cat:
    sns.catplot(x=i, y="P. ESI", data=examine,height=4, aspect=4,kind="box")
```

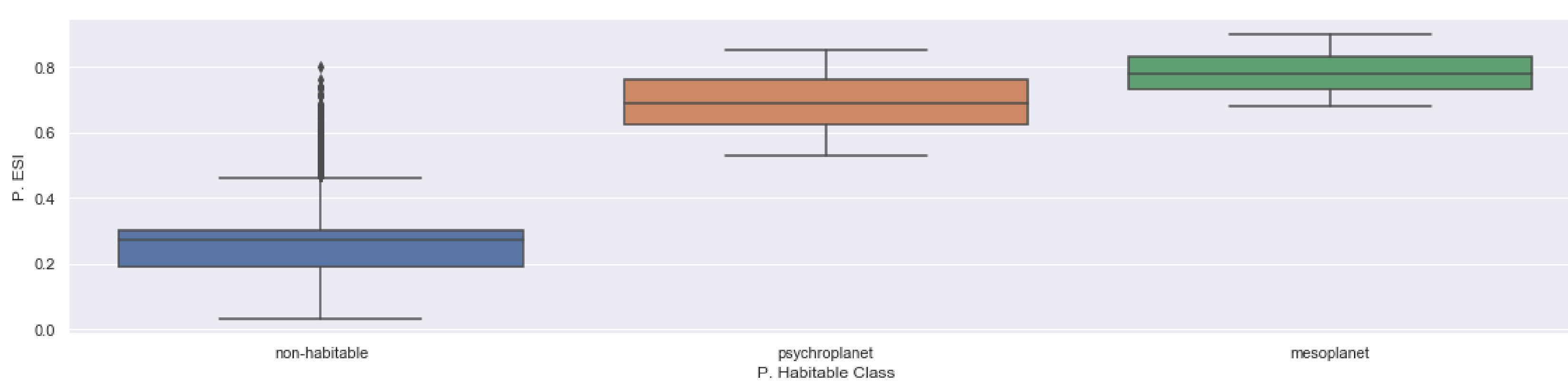
```
<Figure size 1440x360 with 0 Axes>
```

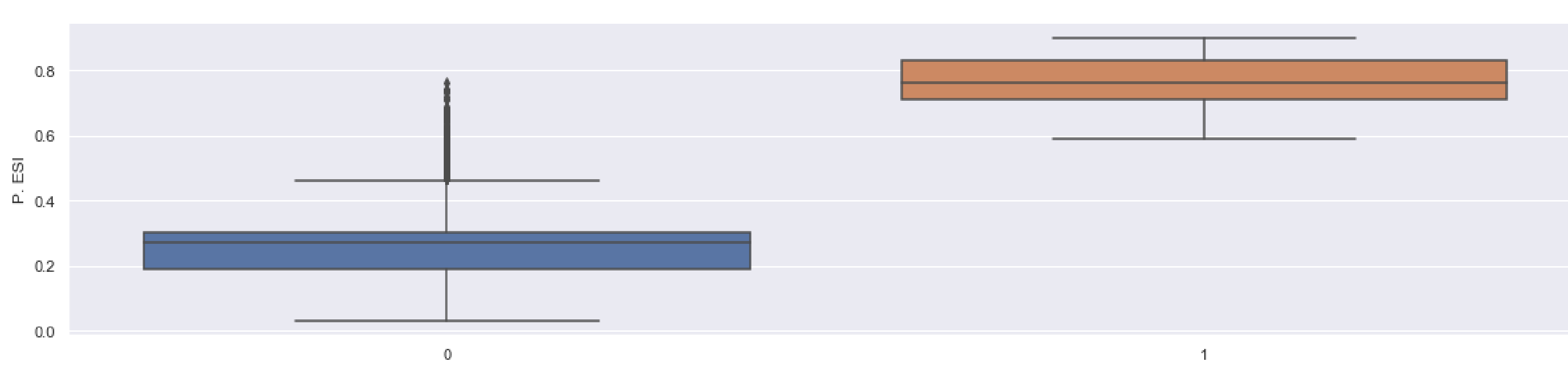






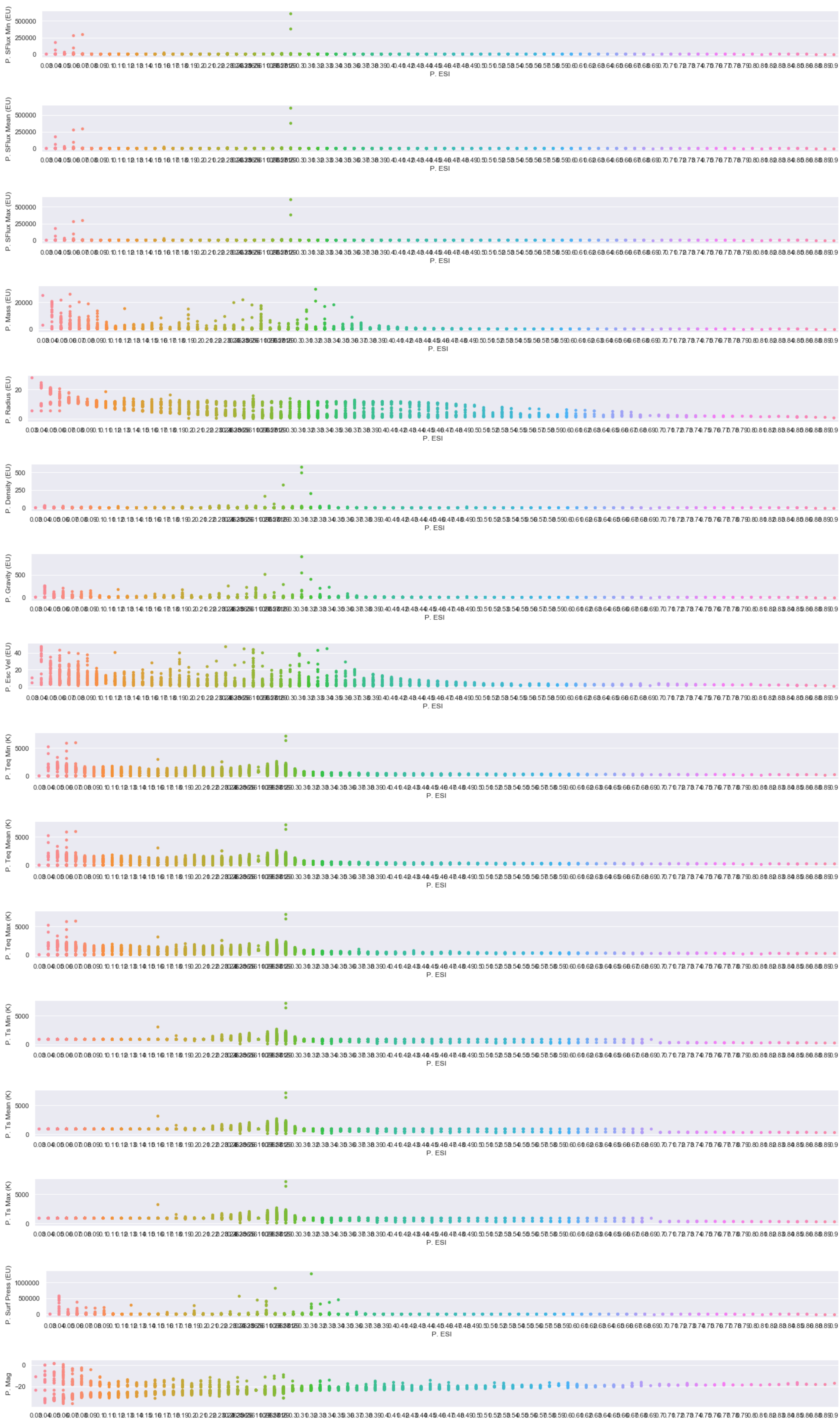


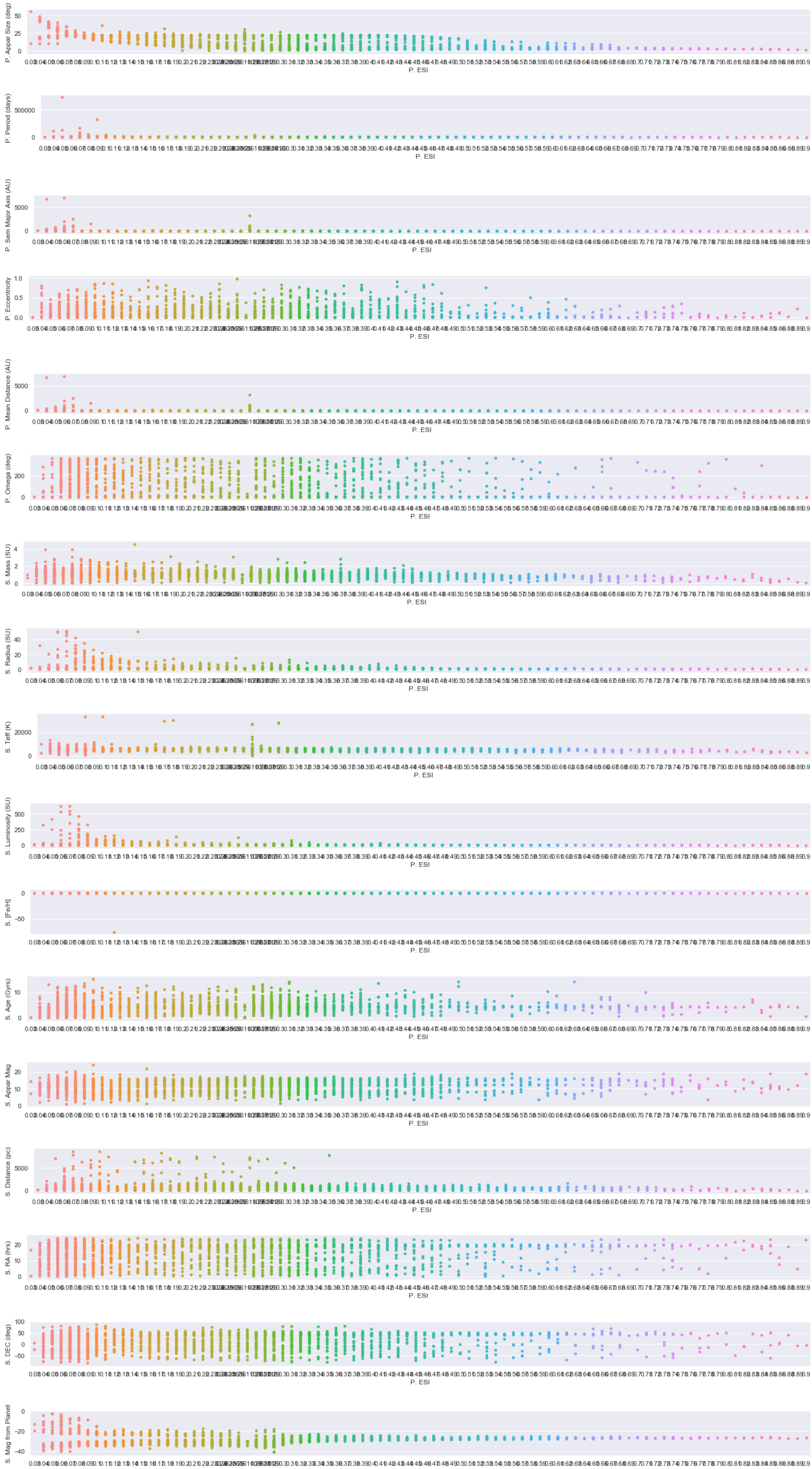


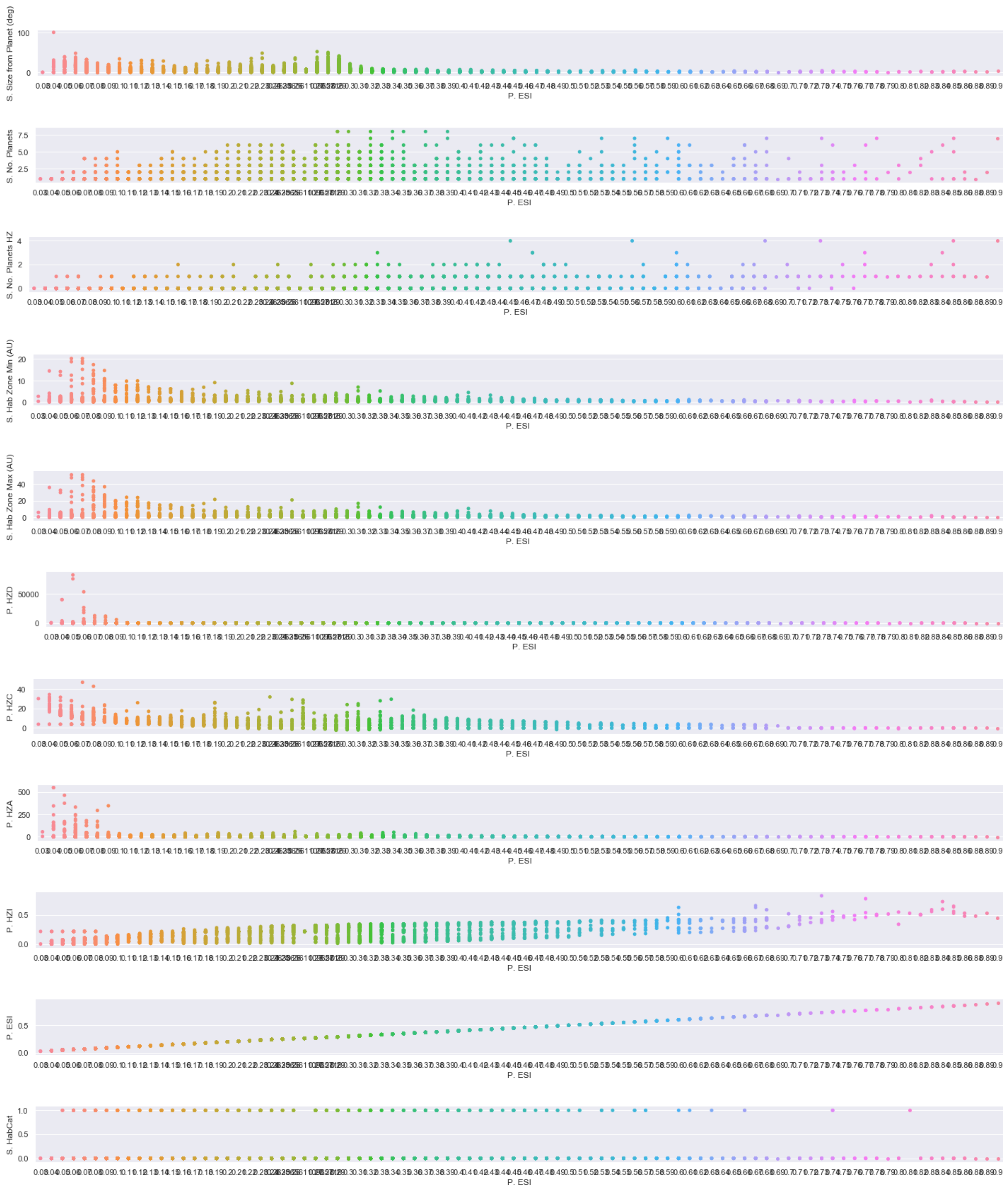


In [35]:
```
for i in pred_num:
    sns.catplot(x='P. ESI', y=i, jitter=False, data=examine,height=2,aspect=10,sharex=True);
```

```
C:\Users\User\Anaconda3\lib\site-packages\matplotlib\pyplot.py:537: RuntimeWarning: More than 20 figures have
been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until exp
licitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_wa
rning`).
  max_open_warning, RuntimeWarning)
```




P. Appar Size (deg)
P. Period (days)
P. Sem Major Axis (AU)
P. Eccentricity
P. Mean Distance (AU)
P. Omega (deg)
S. Mass (SU)
S. Radius (SU)
S. Teff (K)
S. Luminosity (SU)
S. [Fe/H]
S. Age (Gyrs)
S. Appar Mag
S. Distance (pc)
S. RA (hrs)
S. DEC (deg)
S. Mag from Planet
P. ESI



Encoding the data

In [36]:
```
X.shape
```

Out[36]: (3869, 49)

In [37]:
```
# Encoding the Independent Variable
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

# Encoding the Dependent Variable
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

# Encoding the Independent Variable
X = pd.concat([X,pd.get_dummies(X['P. Zone Class'], prefix='P. Zone Class',drop_first=True)],axis=1)
X.drop(['P. Zone Class'],axis=1, inplace=True)

X = pd.concat([X,pd.get_dummies(X['P. Mass Class'], prefix='P. Mass Class',drop_first=True)],axis=1)
X.drop(['P. Mass Class'],axis=1, inplace=True)

X = pd.concat([X,pd.get_dummies(X['P. Composition Class'], prefix='P. Composition Class',drop_first=True)],ax
is=1)
X.drop(['P. Composition Class'],axis=1, inplace=True)

X = pd.concat([X,pd.get_dummies(X['P. Atmosphere Class'], prefix='P. Atmosphere Class',drop_first=True)],axis
=1)
X.drop(['P. Atmosphere Class'],axis=1, inplace=True)
```

In [38]:
```
print(X.shape)
print(X.columns)
X.head()
```

```
(3869, 58)
Index(['P. Habitable', 'P. SFlux Min (EU)', 'P. SFlux Mean (EU)',
       'P. SFlux Max (EU)', 'P. Mass (EU)', 'P. Radius (EU)',
       'P. Density (EU)', 'P. Gravity (EU)', 'P. Esc Vel (EU)',
       'P. Teq Min (K)', 'P. Teq Mean (K)', 'P. Teq Max (K)', 'P. Ts Min (K)',
       'P. Ts Mean (K)', 'P. Ts Max (K)', 'P. Surf Press (EU)', 'P. Mag',
       'P. Appar Size (deg)', 'P. Period (days)', 'P. Sem Major Axis (AU)',
       'P. Eccentricity', 'P. Mean Distance (AU)', 'P. Omega (deg)',
       'S. Mass (SU)', 'S. Radius (SU)', 'S. Teff (K)', 'S. Luminosity (SU)',
       'S. [Fe/H]', 'S. Age (Gyrs)', 'S. Appar Mag', 'S. Distance (pc)',
       'S. RA (hrs)', 'S. DEC (deg)', 'S. Mag from Planet',
       'S. Size from Planet (deg)', 'S. No. Planets', 'S. No. Planets HZ',
       'S. Hab Zone Min (AU)', 'S. Hab Zone Max (AU)', 'P. HZD', 'P. HZC',
       'P. HZA', 'P. HZI', 'P. ESI', 'S. HabCat', 'P. Zone Class_Hot',
       'P. Zone Class_Warm', 'P. Mass Class_Mercurian',
       'P. Mass Class_Neptunian', 'P. Mass Class_Subterran',
       'P. Mass Class_Superterran', 'P. Mass Class_Terran',
       'P. Composition Class_iron', 'P. Composition Class_rocky-iron',
       'P. Composition Class_rocky-water', 'P. Composition Class_water-gas',
       'P. Atmosphere Class_metals-rich', 'P. Atmosphere Class_no-atmosphere'],
      dtype='object')
```

Out[38]:

	P. Habitable	P. SFlux Min (EU)	P. SFlux Mean (EU)	P. SFlux Max (EU)	P. Mass (EU)	P. Radius (EU)	P. Density (EU)	P. Gravity (EU)	P. Esc Vel (EU)	P. Teq Min (K)	...	P. Mass Class_Neptunian	Class_S
0	0	4.080000e-06	4.080000e-06	4.080000e-06	4451.16	19.04	0.64	12.28	15.29	11.4	...	0	0
1	0	2.166914e-02	2.166914e-02	2.166914e-02	6358.80	10.94	4.86	53.12	24.11	97.7	...	0	0
2	0	3.960000e-06	3.960000e-06	3.960000e-06	4133.22	11.40	2.79	31.79	19.04	11.4	...	0	0
3	0	1.030000e-05	1.030000e-05	1.030000e-05	6358.80	11.20	4.53	50.69	23.83	14.4	...	0	0
4	0	2.370000e-07	2.370000e-07	2.370000e-07	4419.37	16.13	1.05	16.99	16.55	5.6	...	0	0

5 rows × 58 columns

DF to ndArr conversion of X

```
(I find it easier to perform the conversion after encoding the categorical variables)
```

In [39]:
```
X = X.values
```

In [40]:
```
X.shape
```

Out[40]: (3869, 58)

In [41]:
```
print("X is: \n\n" + repr(X))
print("y is: \n\n" + repr(y))
```

```
X is:

array([[0.000000e+00, 4.080000e-06, 4.080000e-06, ..., 0.000000e+00,
        0.000000e+00, 0.000000e+00],
       [0.000000e+00, 2.166914e-02, 2.166914e-02, ..., 0.000000e+00,
        0.000000e+00, 0.000000e+00],
       [0.000000e+00, 3.960000e-06, 3.960000e-06, ..., 0.000000e+00,
        0.000000e+00, 0.000000e+00],
       ...,
       [0.000000e+00, 9.110547e+00, 9.110548e+00, ..., 0.000000e+00,
        1.000000e+00, 0.000000e+00],
       [0.000000e+00, 4.674975e+00, 5.060307e+00, ..., 0.000000e+00,
        1.000000e+00, 0.000000e+00],
       [0.000000e+00, 2.268338e+00, 2.915344e+00, ..., 0.000000e+00,
        1.000000e+00, 0.000000e+00]])
y is:

array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

Feature Scaling

In [42]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

In [43]:
```
print("X is: \n\n" + repr(X))
print("y is: \n\n" + repr(y))
```

```
X is:

array([[-0.11325734, -0.05969398, -0.0606261 , ..., -0.07900552,
        -1.31691899, -0.32857427],
       [-0.11325734, -0.05969239, -0.06062451, ..., -0.07900552,
        -1.31691899, -0.32857427],
       [-0.11325734, -0.05969398, -0.0606261 , ..., -0.07900552,
        -1.31691899, -0.32857427],
       ...,
       [-0.11325734, -0.05902641, -0.05995858, ..., -0.07900552,
         0.75934815, -0.32857427],
       [-0.11325734, -0.05935142, -0.06025533, ..., -0.07900552,
         0.75934815, -0.32857427],
       [-0.11325734, -0.05952777, -0.06041249, ..., -0.07900552,
         0.75934815, -0.32857427]])
y is:

array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

Splitting the dataset into the Training set and Test set

In [44]:
```
from sklearn.cross_validation import train_test_split
#X = X[:,[10,11,24]]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 120)
```

```
C:\Users\User\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This module was
deprecated in version 0.18 in favor of the model_selection module into which all the refactored classes and f
unctions are moved. Also note that the interface of the new CV iterators are different from that of this modu
le. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

Feature Scaling

from sklearn.preprocessing import StandardScaler sc = StandardScaler() X_train = sc.fit_transform(X_train) X_test = sc.transform(X_test)

Fitting Naive Bayes to the Training set

In [45]:
```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Out[45]:
```
GaussianNB(priors=None)
```

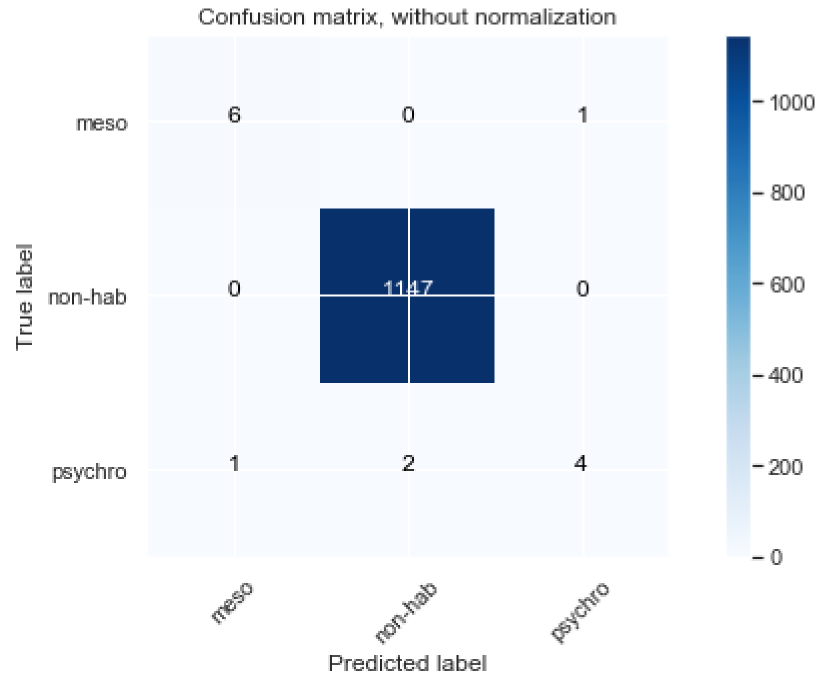
Predicting the Test set results

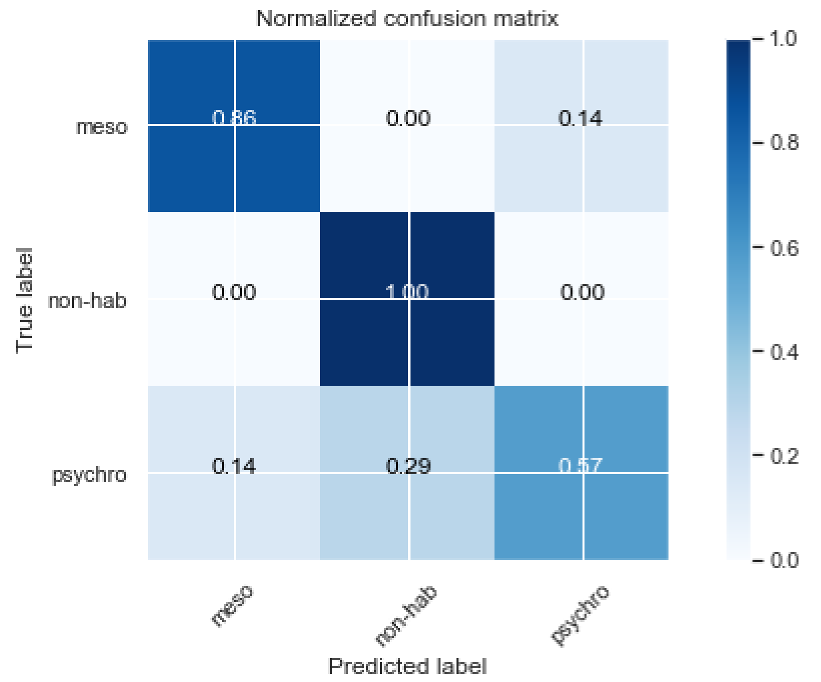
In [46]:
```
y_pred = classifier.predict(X_test)
```

Making the Confusion Matrix

In [47]:
```
import itertools

from sklearn import svm
from sklearn.metrics import confusion_matrix


class_names = ['meso','non-hab','psychro']

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()


# Compute confusion matrix
cnf_matrix_SVM = confusion_matrix(y_test, y_pred)
cnf_matrix = cnf_matrix_SVM

np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(figsize=(8,5))
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix, without normalization')

# Plot normalized confusion matrix
plt.figure(figsize=(8,5))
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize=True,
                      title='Normalized confusion matrix')

plt.show()
```

```
Confusion matrix, without normalization
[[   6    0    1]
 [   0 1147    0]
 [   1    2    4]]
Normalized confusion matrix
[[0.86 0.   0.14]
 [0.   1.   0.  ]
 [0.14 0.29 0.57]]
```





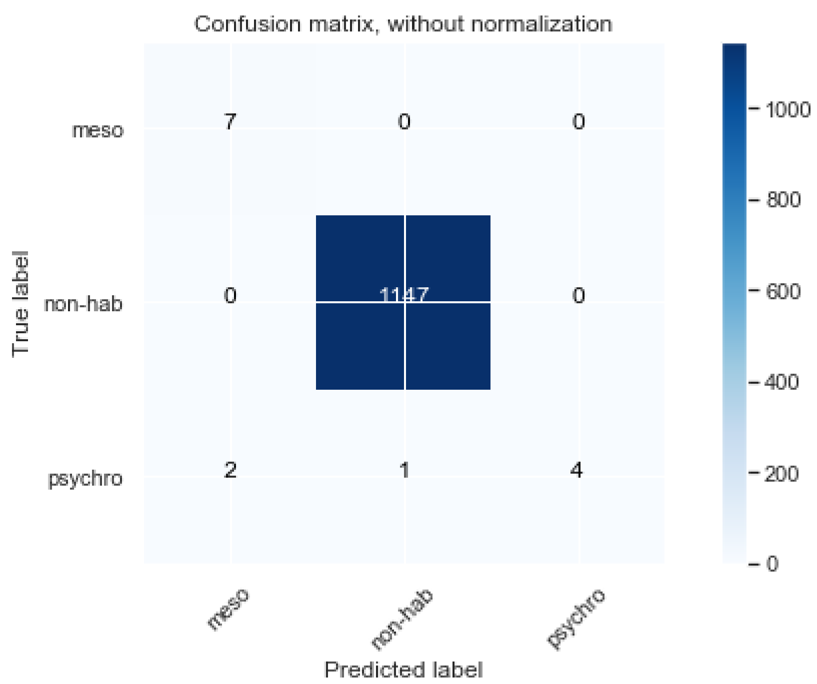
Fitting Decision Tree to the training set

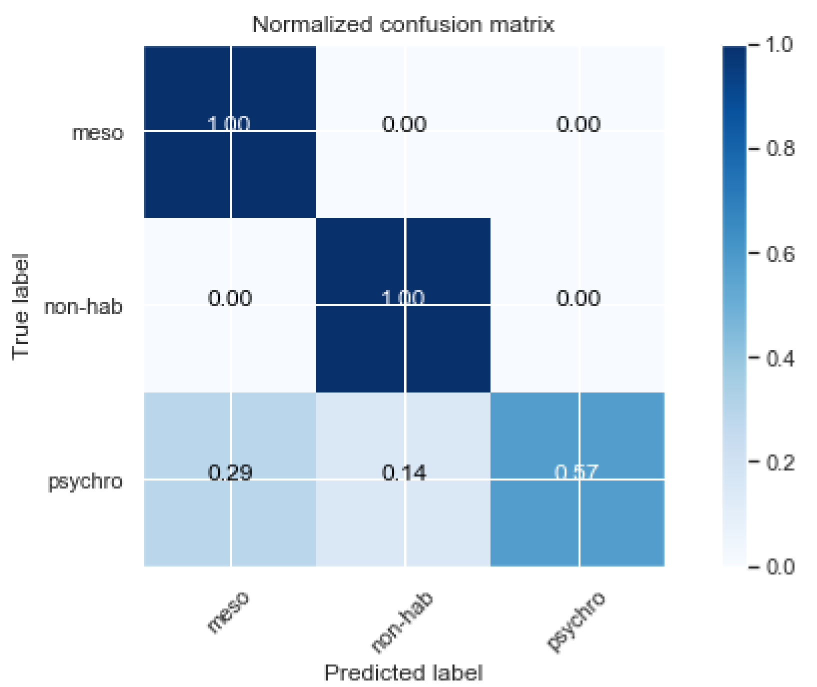
In [48]:
```
# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

Confusion Matrix

In [49]:
```
# Compute confusion matrix
cnf_matrix_DT = confusion_matrix(y_test, y_pred)
cnf_matrix = cnf_matrix_DT
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(figsize=(8,5))
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix, without normalization')

# Plot normalized confusion matrix
plt.figure(figsize=(8,5))
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize=True,
                      title='Normalized confusion matrix')

plt.show()
```

```
Confusion matrix, without normalization
[[   7    0    0]
 [   0 1147    0]
 [   2    1    4]]
Normalized confusion matrix
[[1.   0.   0.  ]
 [0.   1.   0.  ]
 [0.29 0.14 0.57]]
```



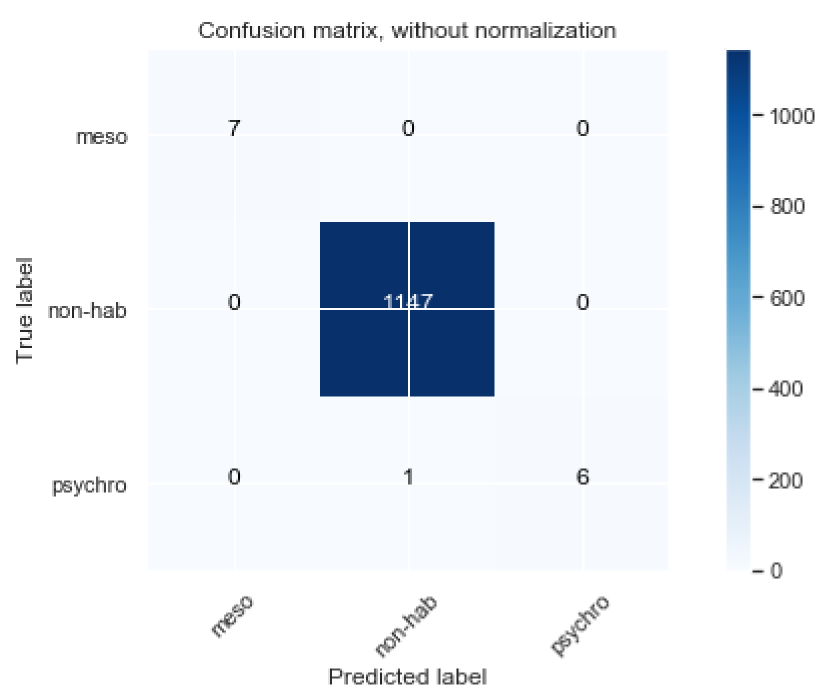


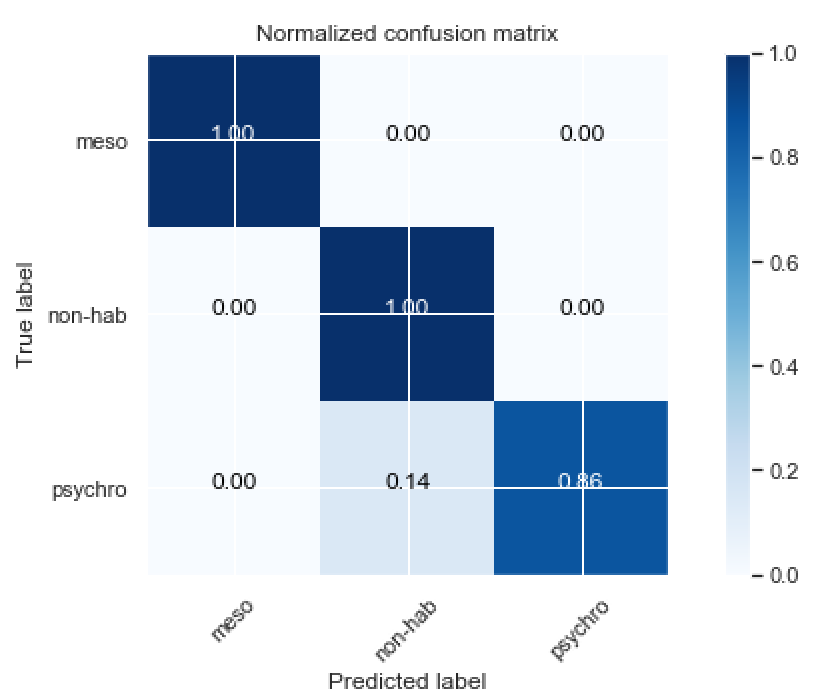
Fitting Random Forest to the training set

In [50]:
```
# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

In [51]:
```
# Compute confusion matrix
cnf_matrix_RF = confusion_matrix(y_test, y_pred)
cnf_matrix = cnf_matrix_RF
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(figsize=(8,5))
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix, without normalization')

# Plot normalized confusion matrix
plt.figure(figsize=(8,5))
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize=True,
                      title='Normalized confusion matrix')

plt.show()
```

```
Confusion matrix, without normalization
[[   7    0    0]
 [   0 1147    0]
 [   0    1    6]]
Normalized confusion matrix
[[1.   0.   0.  ]
 [0.   1.   0.  ]
 [0.   0.14 0.86]]
```



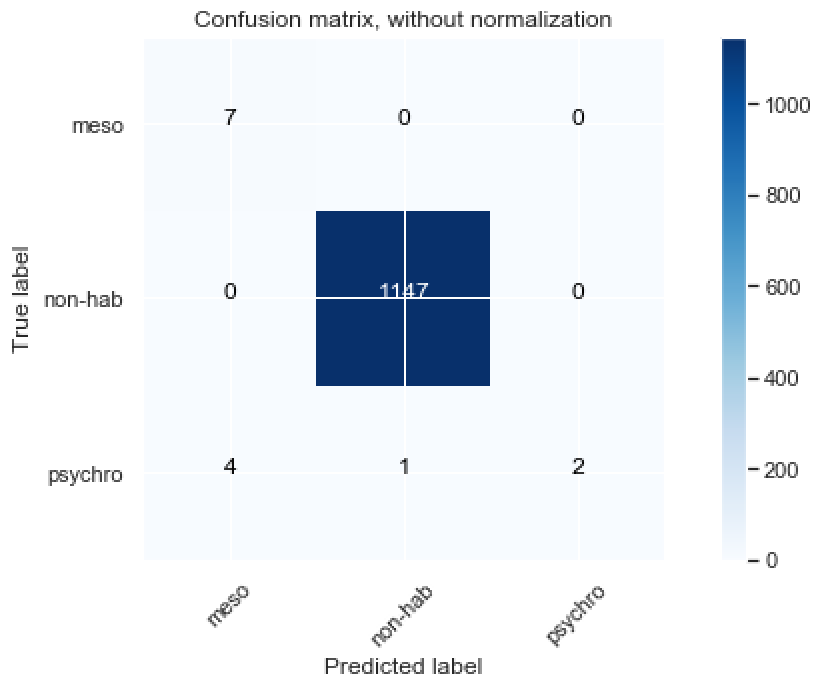


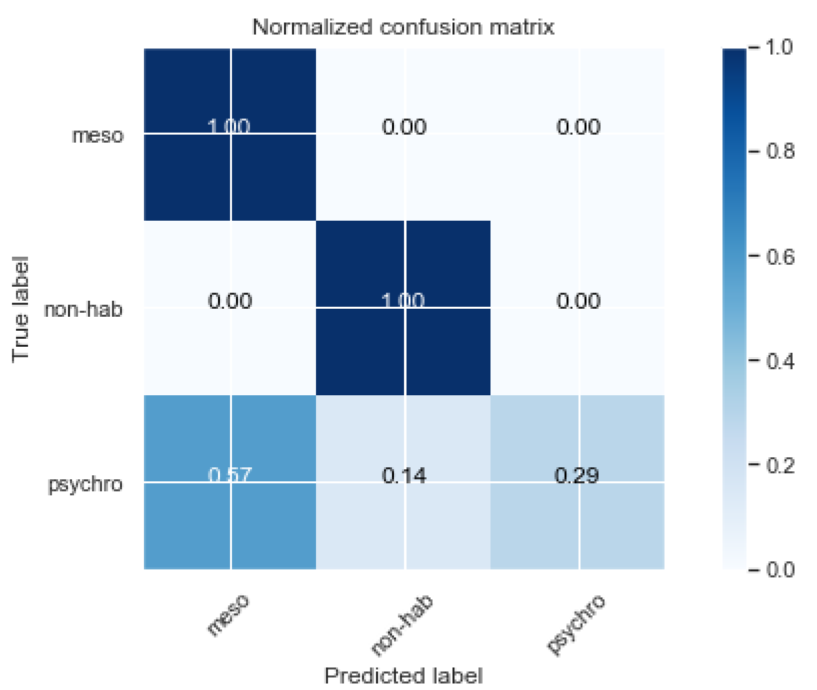
Fitting K-nearest Neighbours to the training set

In [52]:
```
# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

In [53]:
```
# Compute confusion matrix
cnf_matrix_KNN = confusion_matrix(y_test, y_pred)
cnf_matrix = cnf_matrix_KNN
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(figsize=(8,5))
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix, without normalization')

# Plot normalized confusion matrix
plt.figure(figsize=(8,5))
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize=True,
                      title='Normalized confusion matrix')

plt.show()
```

```
Confusion matrix, without normalization
[[   7    0    0]
 [   0 1147    0]
 [   4    1    2]]
Normalized confusion matrix
[[1.   0.   0.  ]
 [0.   1.   0.  ]
 [0.57 0.14 0.29]]
```





NOTE

1. Since we have find only a few habitable planets, our dataset is heavily dominated by the Non-Habitable class.
2. High accuracy is mostly because of Non-Habitable class dominance which is influencing the rate of false positive predictions.
3. Combining other detection methods like Image Classification, Time Series Data Classification,Luminosity Calculation can make this project better by providing us better accuracy by reducing the number of false positives in our prediction.
4. Using different algorithms can also reduce the percentage of errors in our project.

Appendix:

[1] RAW Data Source: http://phl.upr.edu/projects/habitable-exoplanets-catalog/data/database (http://phl.upr.edu/projects/habitable-exoplanets-catalog/data/database) PHL's Exoplanets Catalog Last Update: July 2, 2018 Introduction The PHL's Exoplanets Catalog (PHL-EC) contains observed and modeled parameters for all currently confirmed exoplanets from the Extrasolar Planets Encyclopedia and NASA Kepler candidates from the NASA Exoplanet Archive, including those potentially habitable. It also contains a few still unconfirmed exoplanets of interest. The main difference between PHL-EC and other exoplanets databases is that it contains more estimated stellar and planetary parameters, habitability assessments with various habitability metrics, planetary classifications, and many corrections. Some interesting inclusions are the identification of those stars in the Catalog of Nearby Habitable Systems (HabCat), the apparent size and brightness of stars and planets as seen from a vantage point (i.e. moon-Earth distance), and the location constellation of each planet.

[2] A Comparative Study in Classification Methods of Exoplanets: Machine Learning Exploration via Mining and Automatic Labeling of the Habitability Catalog Surbhi Agrawal1 ?, Suryoday Basak1 , Snehanshu Saha1†, Marialis Rosario-Franco2 , Swati Routh3 , Kakoli Bora4 , Abhijit Jeremiel Theophilus1 1Department of Computer Science and Engineering, PESIT Bangalore South Campus, Karnataka, India 560100 2Physics Department, University of Texas at Arlington 3Physics Department, CPGS, Jain University 4Department of Information Science and Engineering, PESIT Bangalore South Campus, Karnataka, India 560100

[3] Emergence of a Habitable Planet - https://link.springer.com/article/10.1007/s11214-007-9225-z (https://link.springer.com/article/10.1007/s11214-007-9225-z)

[4] Honorary Mention: http://curious.astro.cornell.edu/ (http://curious.astro.cornell.edu/)

This project can be found at my portfolio website www.kaustavbasu.me/minor-project-fifth-sem

END