EXOPLANETS

Final Project

Claire Oneglia Russeil

Exoplanet

Any planet located outside our solar system

- 1995 : thanks to new technology, the first exoplanets were discovered.
- From 1995 to 2023, thousands of exoplanets have been identified and cataloged (more than 6.000 confirmed, 15.000 non-confirmed).
- Two space telescopes have been launched, as tool for the study of exoplanets (Hubble in 1990, James Webb in 2021).
- Like any other planet, an exoplanet orbits around a star.

NASA Exoplanet Archive

NASA Exoplanet Science Institute

Home | About Us | Data | Tools | Support | Login

Select Columns | Download Table | Plot Table | View Documentation | User Preferences

Planetary Systems

	Planet Name	Host Name	Default Parameter Set	Number of Stars	Number of Planets	Discovery Method	Discovery Year	Discovery Facility	Solution Type	Controversial Flag	Planetary Pa
☑	11 Com b	11 Com	0	2	1	Radial Velocity	2007	Xinglong Station	Published Confirmed	0	Kunitomo et al.
☑	11 Com b	11 Com	0	2	1	Radial Velocity	2007	Xinglong Station	Published Confirmed	0	Liu et al. 2008
☑	11 Com b	11 Com	1	2	1	Radial Velocity	2007	Xinglong Station	Published Confirmed	0	Teng et al. 202
☑	11 UMi b	11 UMi	1	1	1	Radial Velocity	2009	Thueringer Lande	Published Confirmed	0	Stassun et al. 2
☑	11 UMi b	11 UMi	0	1	1	Radial Velocity	2009	Thueringer Lande	Published Confirmed	0	Kunitomo et al.
☑	11 UMi b	11 UMi	0	1	1	Radial Velocity	20				Dollinger et al.
☑	14 And b	14 And	1	1	1	Radial Velocity	20				Teng et al. 202
☑	14 And b	14 And	0	1	1	Radial Velocity	20				Kunitomo et al.
☑	14 And b	14 And	0	1	1	Radial Velocity	20				Sato et al. 2008
☑	14 Her b	14 Her	0	1	2	Radia	20				Wittenmyer et a
☑	14 Her b	14 Her	0	1	2	Radial Velocity	20				Gozdziewski et
☑	14 Her b	14 Her	1	1	2	Radial Velocity	20				Feng et al. 202
☑	14 Her b	14 Her	0	1	2	Radial Velocity	20				Stassun et al. 2
☑	14 Her b	14 Her	0	1	2	Radial Velocity	20				Naef et al. 2004
☑	14 Her b	14 Her	0	1	2	Radial Velocity	20				Rosenthal et al.
☑	14 Her b	14 Her	0	1	2	Radial Velocity	2002	W. M. Keck Obser	Published Confirmed	0	Butler et al. 200
☑	14 Her b	14 Her	0	1	2	Radial Velocity	2002	W. M. Keck Obser	Published Confirmed	0	Gozdziewski et

```python
exoplanet.head()
```

	sy_snum	sy_pnum	disc_year	pl_orbper	pl_orbsmax	pl_bmass
0	1	1	1995	4.231000	0.052000	146.2018
1	1	1	1995	4.230785	0.052700	150.0090
2	1	1	1995	4.230797	0.052350	147.4723
3	1	3	1996	1076.600000	2.100000	778.6835
4	2	5	1996	14.651520	0.115227	263.9785

```python
exoplanet.shape
```

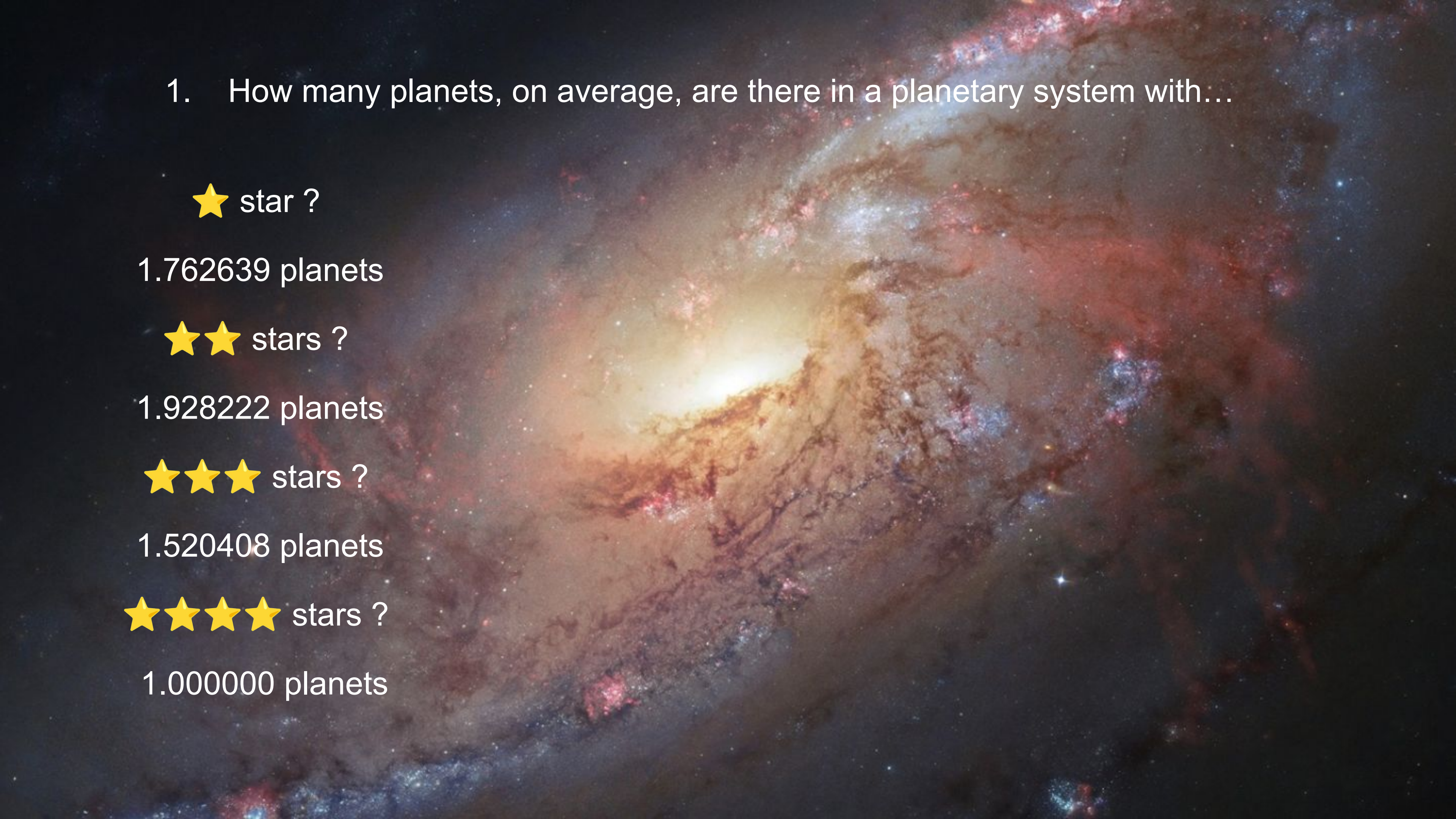
```
(3505, 14)
```

COLUMNS MEANING

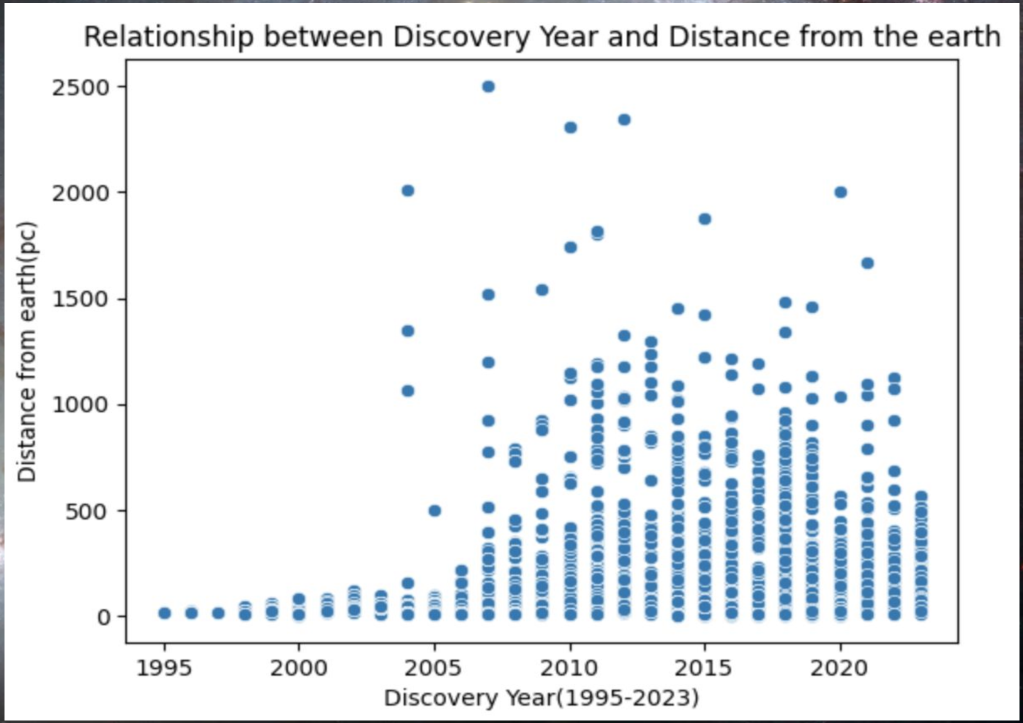
- Col 1. sy_snum : Number of Stars: Number of stars in the planetary system of the exoplanet described in the line.
- Col 2. sy_pnum : Number of Planets: Number of planets in the planetary system of the exoplanet described in the line.
- Col 3. disc_year : Discovery Year
- Col 4. pl_orbper : Orbital Period [days]: Duration in days of the planet's orbit (for Earth, it would be 365, for example).
- Col 5. pl_orbsmax : Orbit Semi-Major Axis [au]): Orbital radius in astronomical units. It is the distance from a planet to its star. For Earth, this would be 1.
- Col 6. pl_bmasse : Planet Mass or Mass*sin(i) [Earth Mass]:Mass of the planet in terms of Earth's mass.
- Col 7. pl_orbeccen : Eccentricity: Parameter that indicates the flattening of the orbit (whether it is a circle or an ellipse).
- Col 8. st_mass : Stellar Mass [Solar mass]: Mass of the planet relative to the mass of its star.
- Col 9. ra : RA [deg]: Coordinates 1 in the sky for the planet.
- Col 10. dec : Dec [deg]: Coordinates 2 in the sky for the planet.
- Col 11. sy_dist : Distance [pc]: Distance of the planet relative to us (in parsecs, with 1 parsec = 3.26 light-years).
- Col 12. sy_vmag : V (Johnson) Magnitude: Luminosity of the star in the green.
- Col 13. sy_kmag : Ks (2MASS) Magnitude: Luminosity of the star in the infrared.
- Col 14. sy_gaiamag : Gaia Magnitude: Luminosity of the star in the red.

QUESTIONS

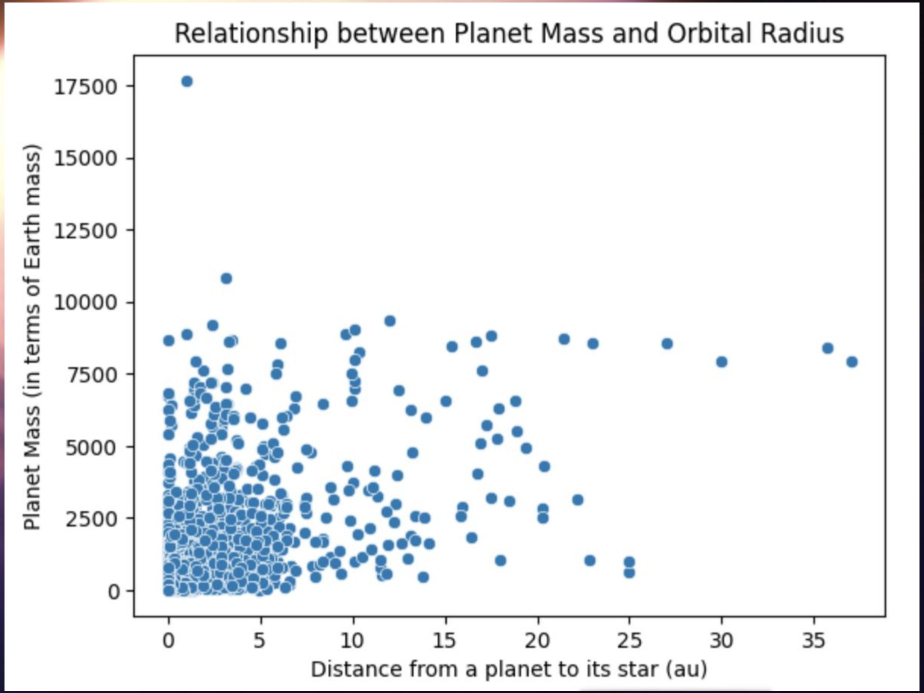
- How many planets, on average, are there in a planetary system with one star ?
- Has the distance of planets (relative to earth) increased from 1995 to nowadays ?
- How does the mass of a planet vary based on its distance from its star ?
- What is the mass of a planet based on the duration of its orbit ?
- What is the duration of a planet’s orbit around its star ?


1. How many planets, on average, are there in a planetary system with…
⭐ star ?
1.762639 planets
⭐⭐ stars ?
1.928222 planets
⭐⭐⭐ stars ?
1.520408 planets
⭐⭐⭐⭐ stars ?
1.000000 planets

2. Has the distance of planets (relative to earth) increased from 1995 to nowadays ?

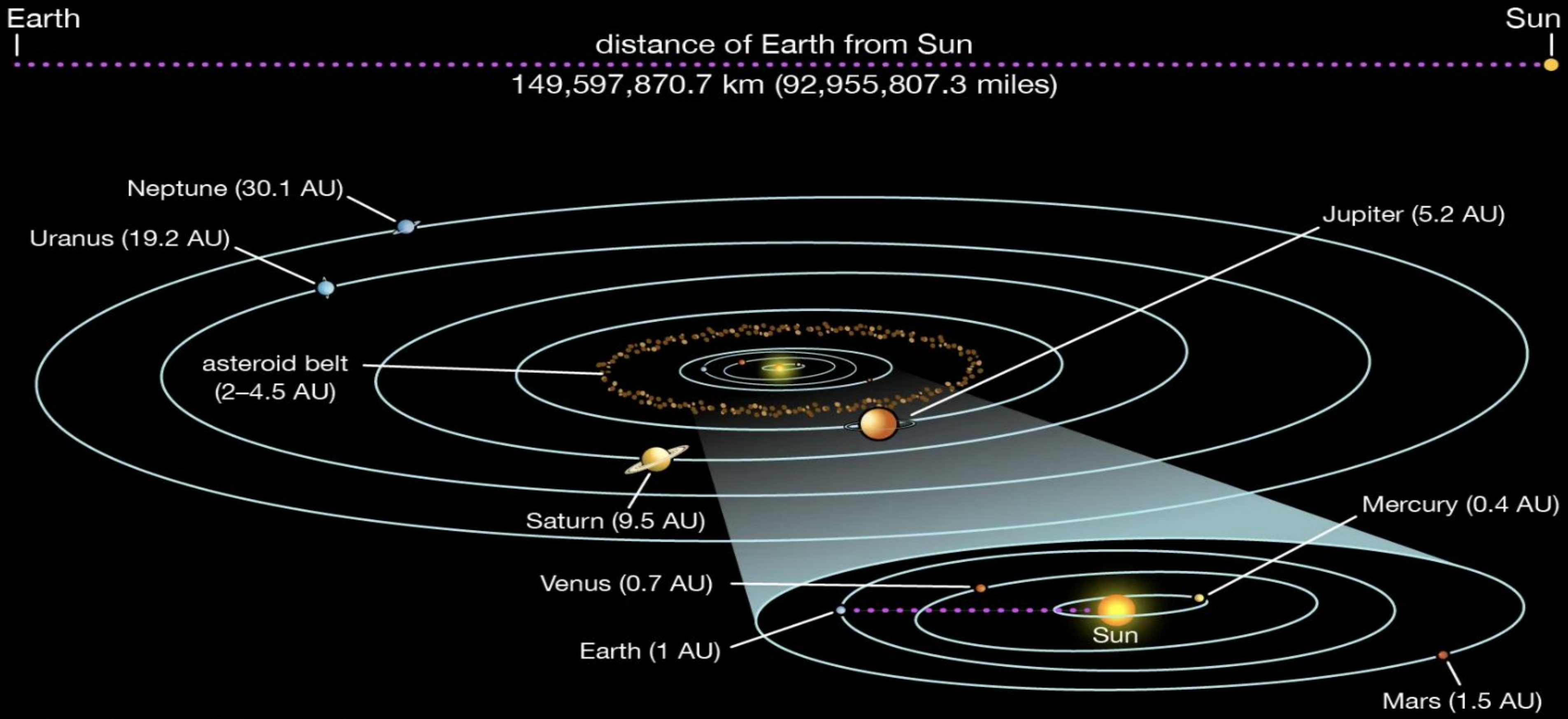


3. How does the mass of a planet vary based on its distance from its star ?
= Orbital Radius in astronomical units

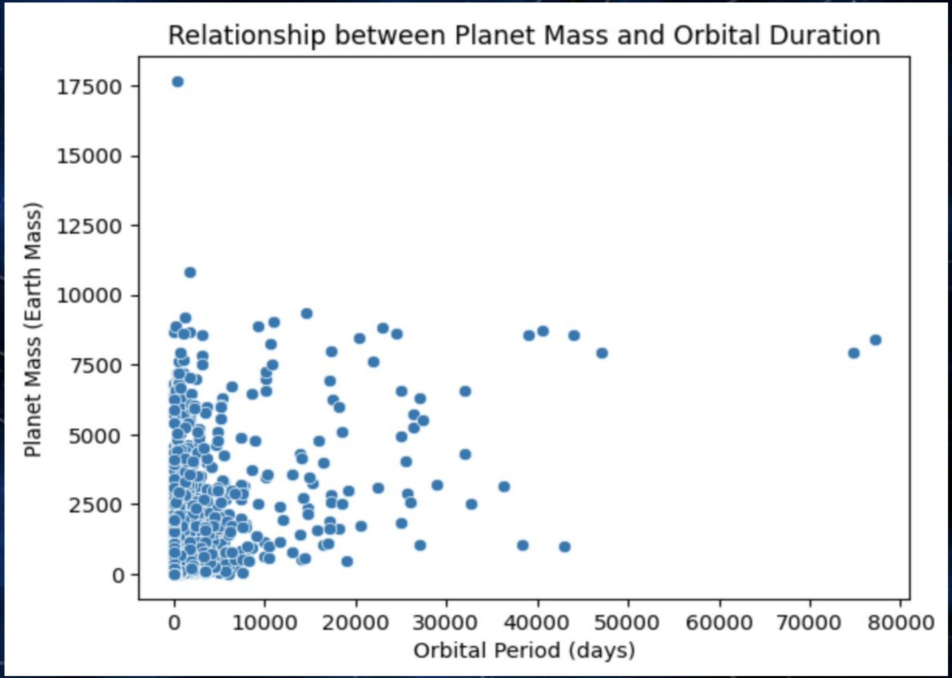


Astronomical unit (AU, or au)

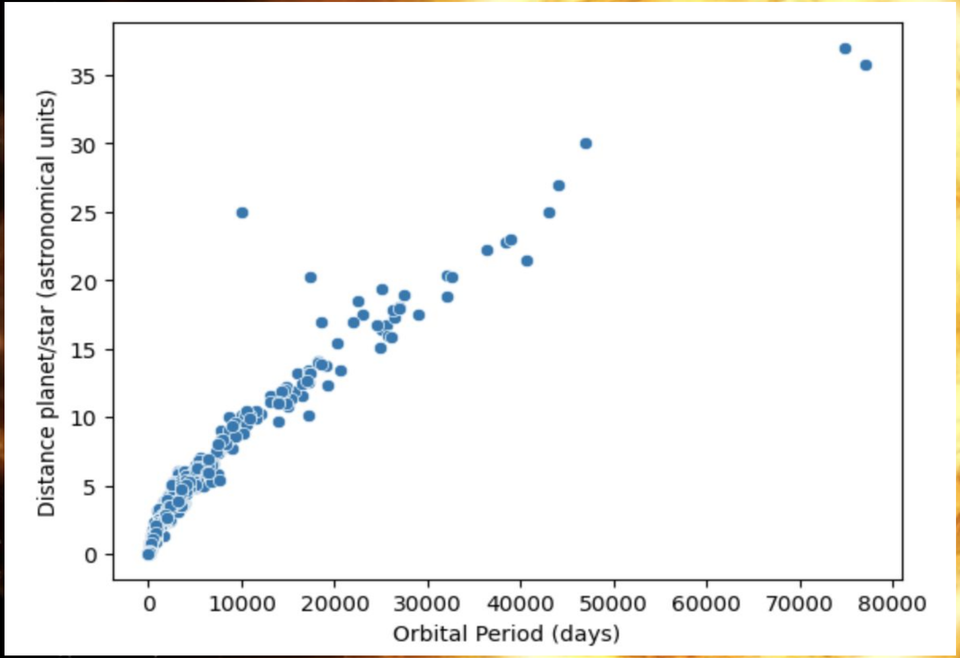
A unit of length effectively equal to the average, or mean, distance between Earth and the Sun.



4. What is the mass of a planet based on the duration of its orbit ?



5. What is the duration of a planet’s orbit around its star ?



Three Regression Models

Linear Regression, Random Forest Regressor, Gradient Boosting Regressor

Models generation 1 :

Features X = column ‘pl_orbper’ + Polynomial Features

Column ‘pl_orbper’ : Orbital Period in days. Duration of the planet orbit (for Earth: 365)

Polynomial Features : the column has been enhanced with features such as itself, its square, its cube

Target y = column ‘pl_orbsmax’ : Distance from a planet to its star, in astronomical units (for Earth: 1)

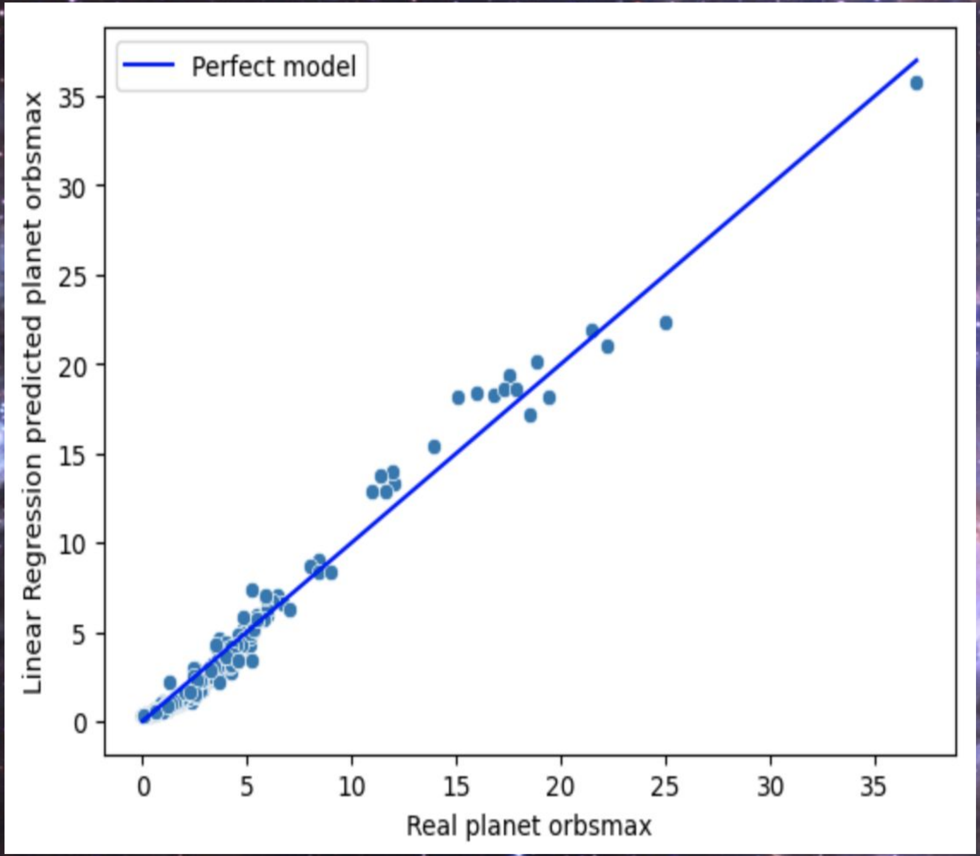
Model generation 2 :

Features X = columns ‘pl_orbsmax’, ‘pl_bmasse’, ‘st_mass’

Column ‘pl_orbsmax’ : Distance from a planet to its star, in astronomical units (for Earth: 1)
Column ‘pl_bmasse’ : Mass of the planet in terms of Earth’s mass
Column ‘st_mass’ : Mass of the planet relative to the mass of its star.

Target y = column ‘pl_orbper’ : Orbital Period in days. Duration of the planet orbit (for Earth: 365)

1. LINEAR REGRESSION



Train

```
print(f"Mean Squared Error (linear_reg_poly Train): {round(mse_train, 2)}")
print(f"R-squared (R2) Score (linear_reg_poly Train): {round(r2_train, 2)}")
```

```
Mean Squared Error (linear_reg_poly Train): 0.34
R-squared (R2) Score (linear_reg_poly Train): 0.95
```

Test

```
mse = mean_squared_error(y_test, y_pred_test_lrp)
print("Mean Squared Error: {:.2f} ".format(mse))
```

```
Mean Squared Error: 0.19
```

```
# Model evaluation using R2 score
r2 = linear_reg_poly.score(X_test_pol_df, y_test)
print(f"R-squared (R2) Score using model.score: {r2}")
```

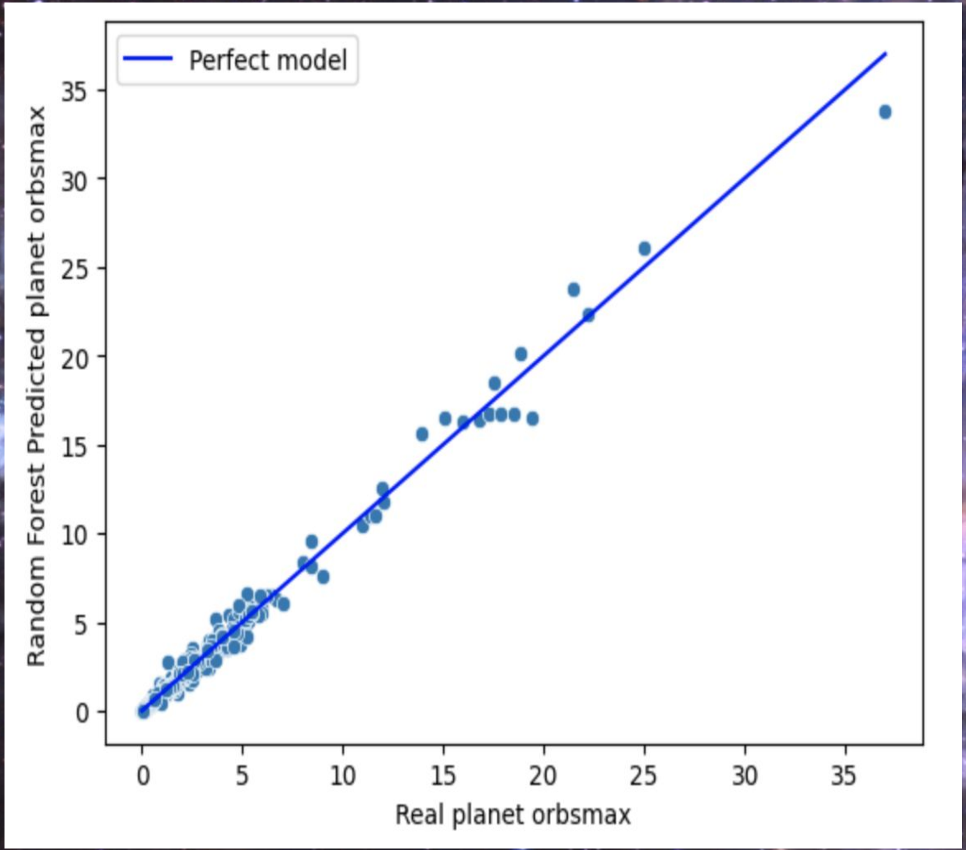
```
R-squared (R2) Score using model.score: 0.9747856437138886
```

Predictions

```
y_pred_test_lrp
```

```
array([0.32802942, 1.94805214, 0.31742163, ..., 3.43143575, 2.86097784,
       1.70015912])
```

2. RANDOM FOREST REGRESSOR



Train

```
print(f"Mean Squared Error (random_forest Train): {round(mse_train, 2)}")
print(f"R-squared (R2) Score (random_forest Train): {round(r2_train, 2)}")
```

```
Mean Squared Error (random_forest Train): 0.04
R-squared (R2) Score (random_forest Train): 0.99
```

Test

```
# Calculation of the Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred_test_rf)
print(f"Mean Squared Error (MSE): {mse}")
```

```
Mean Squared Error (MSE): 0.07746585184451485
```

```
# Calculation of the Root Mean Squared Error
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Root Mean Squared Error (RMSE): 0.2783268794861805
```

```
# Model evaluation using R2 score
r2 = random_forest.score(X_test_pol_df, y_test)
print(f"R-squared (R2) Score using model.score: {r2}")
```

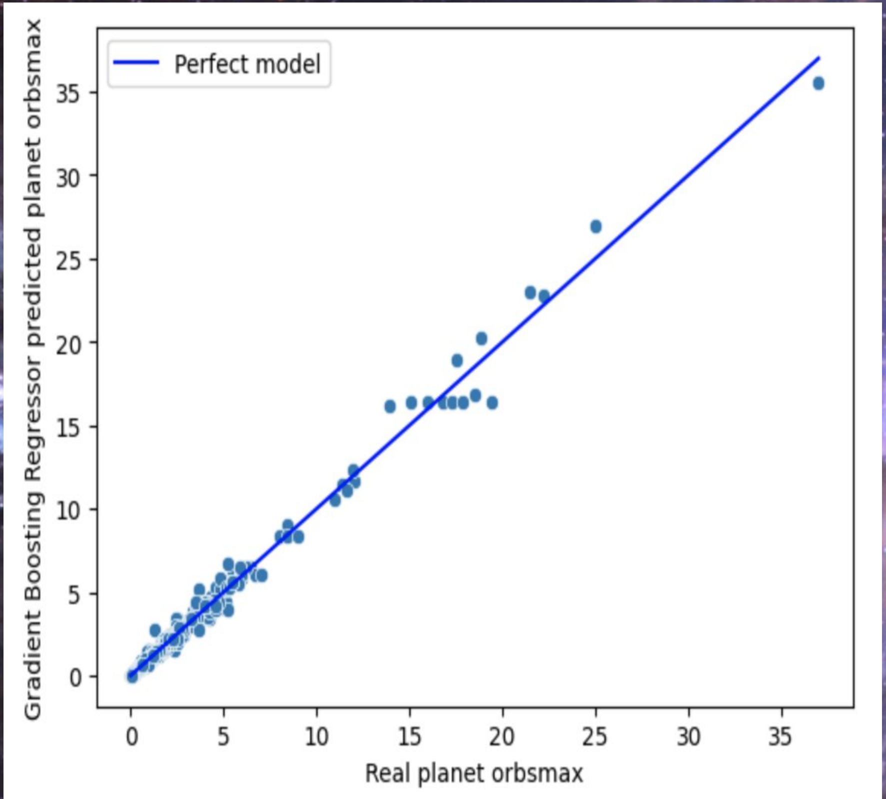
```
R-squared (R2) Score using model.score: 0.9897923345950778
```

Predictions

```
y_pred_test_rf
```

```
array([0.098809 , 2.60578  , 0.0438644, ..., 4.5      , 3.35965  ,
       2.213    ])
```

3. GRADIENT BOOSTING REGRESSOR



Train

```python
print(f"Mean Squared Error (GradientBoostingRegressor Train): {round(mse_train, 2)}")
print(f"R-squared (R2) Score (GradientBoostingRegressor Train): {round(r2_train, 2)}")
```

```
Mean Squared Error (GradientBoostingRegressor Train): 0.03
R-squared (R2) Score (GradientBoostingRegressor Train): 1.0
```

Test

```python
# Calculation of the Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred_test_gbr)
print(f"Mean Squared Error (MSE): {mse}")
```

```
Mean Squared Error (MSE): 0.06443490405201585
```

```python
# Calculation of the Root Mean Squared Error
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Root Mean Squared Error (RMSE): 0.2538403121098299
```

```python
# Model evaluation using R2 score
r2 = gradient_reg.score(X_test_pol_df, y_test)
print(f"R-squared (R2) Score using model.score: {r2}")
```

```
R-squared (R2) Score using model.score: 0.9915094209216029
```

Predictions

```python
y_pred_test_gbr
```

```
array([0.09525114, 2.5795575 , 0.04431979, ..., 4.22946535, 3.40421227,
       2.23155788])
```

Three Regression Models

Linear Regression, Random Forest Regressor, Gradient Boosting Regressor

Models generation 1 :

Features X = column ‘pl_orbper’ + Polynomial Features

Column ‘pl_orbper’ : Orbital Period in days. Duration of the planet orbit (for Earth: 365)

Polynomial Features : the column has been enhanced with features such as itself, its square, its cube

Target y = column ‘pl_orbsmax’ : Distance from a planet to its star, in astronomical units (for Earth: 1)

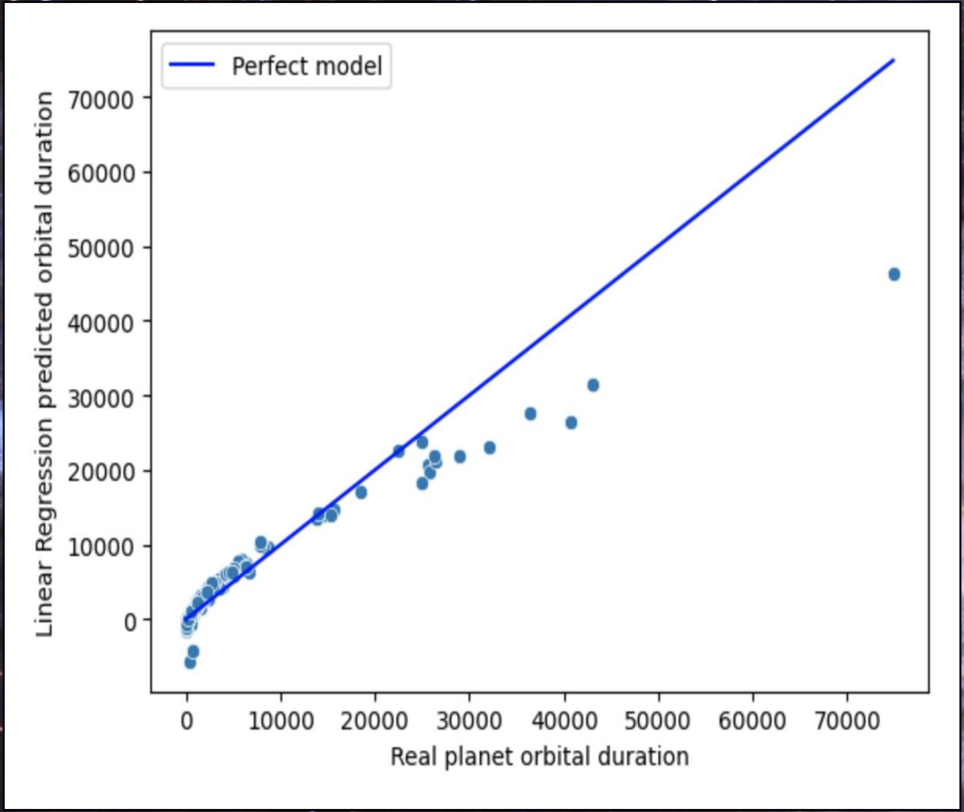
Model generation 2 :

Features X = columns ‘pl_orbsmax’, ‘pl_bmasse’, ‘st_mass’

Column ‘pl_orbsmax’ : Distance from a planet to its star, in astronomical units (for Earth: 1)
Column ‘pl_bmasse’ : Mass of the planet in terms of Earth’s mass
Column ‘st_mass’ : Mass of the planet relative to the mass of its star.

Target y = column ‘pl_orbper’ : Orbital Period in days. Duration of the planet orbit (for Earth: 365)

1. LINEAR REGRESSION



Train

```
print(f"Mean Squared Error (linear_reg Train): {round(mse_train, 2)}")
print(f"R-squared (R2) Score (linear_reg Train): {round(r2_train, 2)}")
```

```
Mean Squared Error (linear_reg Train): 1535714.23
R-squared (R2) Score (linear_reg Train): 0.88
```

Test

```
# Calculation of the Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred_test_lm)
print(f"Mean Squared Error (MSE): {mse}")
```

```
Mean Squared Error (MSE): 2008823.918768723
```

```
# Calculation of the Root Mean Squared Error
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Root Mean Squared Error (RMSE): 1417.3298553155237
```

```
# Model evaluation using R2 score
r2 = linear_reg.score(X_test, y_test)
print(f"R-squared (R2) Score using model.score: {r2}")
```

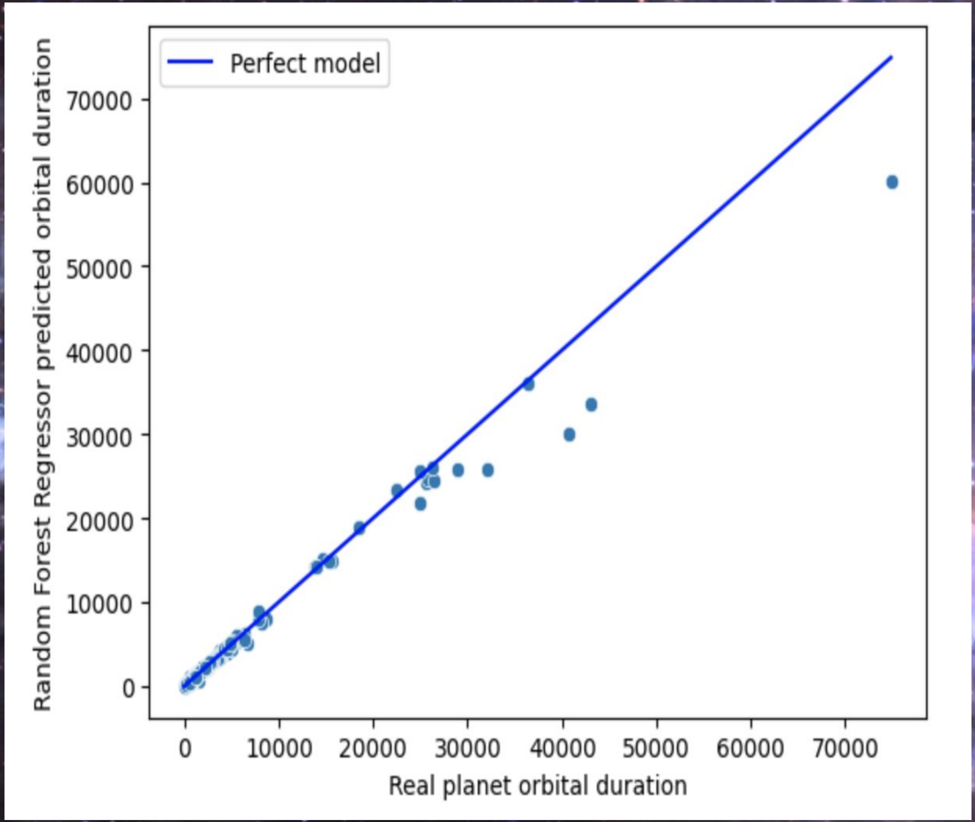
```
R-squared (R2) Score using model.score: 0.8873662429227098
```

Predictions

```
y_pred_test_lm
```

```
array([-375.17552548, 2554.51021791, -668.07362206, ..., 4951.33569183,
        3622.61276195, 2228.94384123])
```

2. RANDOM FOREST REGRESSOR



Train

```
print(f"Mean Squared Error (random_forest_rm Train): {round(mse_train, 2)}")
print(f"R-squared (R2) Score (random_forest_rm Train): {round(r2_train, 2)}")
```

```
Mean Squared Error (random_forest_rm Train): 144430.26
R-squared (R2) Score (random_forest_rm Train): 0.99
```

Test

```
# Calculation of the Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred_test_rfrm)
print(f"Mean Squared Error (MSE): {mse}")
```

```
Mean Squared Error (MSE): 472459.7418615494
```

```
# Calculation of the Root Mean Squared Error
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Root Mean Squared Error (RMSE): 687.3570701328018
```

```
# Model evaluation using R2 score
r2 = random_forest_rm.score(X_test, y_test)
print(f"R-squared (R2) Score using model.score: {r2}")
```

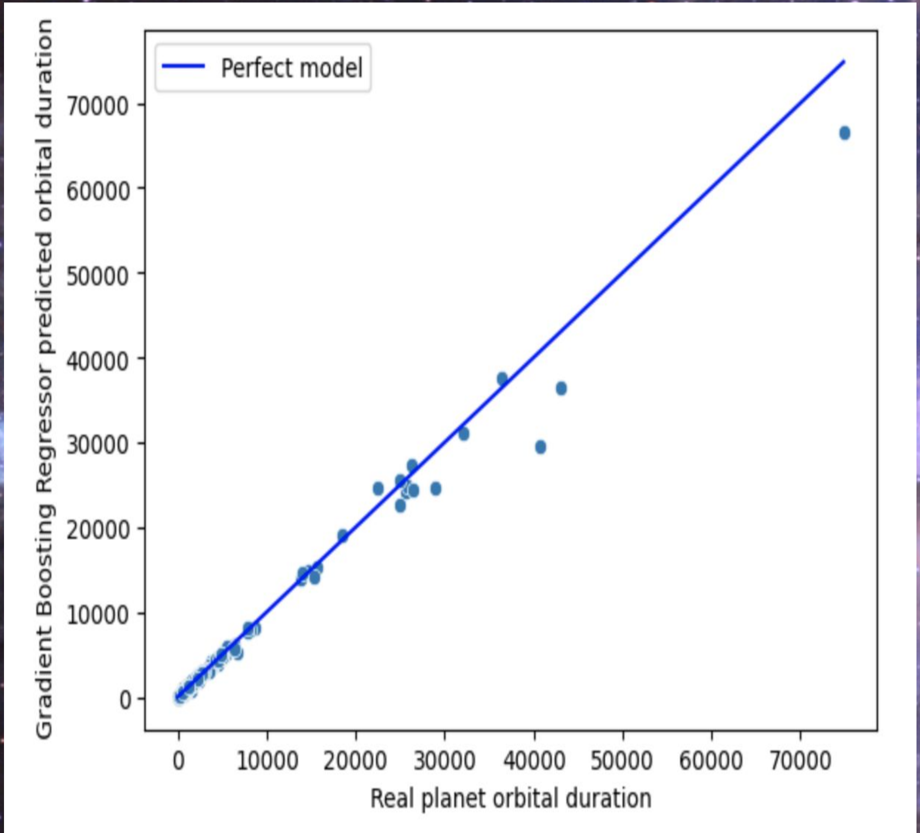
```
R-squared (R2) Score using model.score: 0.9735094174773415
```

Predictions

```
y_pred_test_rfrm
```

```
array([  12.4017014 , 1439.4498401 ,    3.12533995, ..., 2818.8347165 ,
       2192.5516    , 1206.79925166])
```

3. GRADIENT BOOSTING REGRESSOR



Train

```
print(f"Mean Squared Error (gradient_boost_reg Train): {round(mse_train, 2)}")
print(f"R-squared (R2) Score (gradient_boost_reg Train): {round(r2_train, 2)}")
```

```
Mean Squared Error (gradient_boost_reg Train): 144430.26
R-squared (R2) Score (gradient_boost_reg Train): 0.99
```

Test

```
# Calculation of the Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred_test_gbr)
print(f"Mean Squared Error (MSE): {mse}")
```

```
Mean Squared Error (MSE): 273760.2741287627
```

```
# Calculation of the Root Mean Squared Error
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Root Mean Squared Error (RMSE): 523.2210566565175
```

```
# Model evaluation using R2 score
r2 = gradient_boost_reg.score(X_test, y_test)
print(f"R-squared (R2) Score using model.score: {r2}")
```

```
R-squared (R2) Score using model.score: 0.9846503977150317
```

Predictions

```
y_pred_test_gbr
```

```
array([  16.17915185, 1414.73201366,    3.57381513, ..., 2771.79900637,
       2161.00685333, 1178.51941878])
```

THANK YOU !