



系外行星轨道周期预测模型

目录

- 1 梳理题目要求
- 2 思路
- 3 学习笔记
- 4 流程记录
- 5 运行成功截图

梳理题目要求

- 1 下载 **NASA 系外行星数据**，将要求的数据导入 `sklearn.linear_model.LinearRegression` 训练模型
- 2 采用 **均方误差** 评估模型性能，随后 **可视化输出预测结果与真实轨道周期的关系** 以及 **最终的线性方程**
- 3 手动使用 **最小二乘法** 来完成线性回归的任务
- 4 使用 **R检测法** 评估线性拟合的相关性
- 5 **可视化** 预测值 vs. 真实值的散点图
- 6 尝试对数据进行 **对数变换**，观察模型效果是否提升
- 7 解释模型的误差

思路

- 1 访问 NASA 系外行星数据库，获取行星数据保存在 **csv 文件** 中（[什么是 csv 文件？](#)）
- 2 **提取** csv 文件中所需要的数据，将不需要的数据 **剔除**（[如何完成提取和剔除操作？](#)）
- 3 使用提取到的有用数据来 **训练模型**（[如何训练模型？](#)）
- 4 使用 **均方误差（MSE）** 来评估模型的预测性能（[如何评估模型的预测性能？](#)）
- 5 使用 **R检测法** 评估线性拟合的相关性（[如何输出R检测法结果](#)）

- 6 将预测值的真实值的关系 **绘制散点图** ([如何绘图?](#))
- 7 输出最终的线性方程 ([如何导出线性方程参数?](#))
- 8 **手动** 使用最小二乘法来完成线性回归的任务 ([如何使用最小二乘法完成线性回归?](#))
- 9 对数据进行 **对数变换**，重复第二到七步，获得模型效果，进行对比 ([如何进行对数变化?](#))
- 10 **可视化** 预测值 vs. 真实值的散点图，并解释模型的误差

相关知识学习

csv 文件

打开CSV文件 [1](#) [2](#) [3](#)

CSV (Comma Separated Values) 文件是一种纯文本文件，用于存储表格和电子表格信息。内容通常是由文本、数字或日期组成的表格。CSV文件的每一行表示表中的一行，逗号分隔行中的每个单元格 [1](#) [2](#)。

在Microsoft Excel中打开CSV文件

1. **双击CSV文件**: 如果你已经安装了Microsoft Excel，只需双击CSV文件即可在Excel中打开它。双击该文件后，可能会看到一个提示，询问你要使用哪个程序打开该文件，选择Microsoft Excel [1](#)。
2. **通过Excel打开**: 如果你已经在使用Microsoft Excel，可以选择“文件”>“打开”，然后选择CSV文件。如果看不到要打开的文件，可能需要将要打开的类型更改为“文本文件 (*.pm, *.txt, *.csv)”。Excel将在新工作簿中显示数据 [2](#)。
3. **导入到现有工作表**: 在“数据”选项卡的“获取和转换数据”组中，单击“从文本/CSV”。在“导入数据”对话框中，双击要导入的CSV文件，然后单击“导入”。在预览对话框中，你可以选择“加载”将数据直接加载到新工作表中，或选择“加载到”将数据加载到表或现有工作表中¹。

csv 文件可以很好地处理表格类数据，而且具有方便处理的特性

具体处理方法可以使用 pandas 里的 read_csv() 获取数据

```
# 加载数据
```

```
df = pd.read_csv("D:/lianshidaiRecurit/03/PSCompPars_2025.02.09_07.27.02.csv")
```

提取和剔除操作

四、筛选含有特定值的列

同样地，我们也可以筛选含有特定值的列。

```
1 # 筛选城市为"Chicago"的列
2 df_filtered_columns = df[df['City'] == 'Chicago']
3 print(df['City'] == 'Chicago')
4 print("***30")
5 print(df_filtered_columns)
```

上面的代码会筛选出城市为"Chicago"的列，并返回一个新的DataFrame：

```
1 0 False
2 1 False
3 2 True
4 3 False
5 Name: City, dtype: bool
6 *****
7      Name  Age  City
8 2  Charlie  35  Chicago
```

可以使用 `df.read_csv()` 返回的 `DataFrame` 数据类型进行数据的提取

同时发现，下载的文件中存在部分数据缺失，这将导致模型训练出现错误，因而需要将存在缺失数据的地方删去

这里使用 `DataFrame` 的 `.dropna()` 方法进行处理，需要输入的参数 `subset` 为所需检测的列数据名称，返回删除缺失数据之后的 `DataFrame`

最后，将处理好的数据所需的列元素提取到 `x`, `y` 中，方便后续训练模型，这里可以使用

`X = df[['pl_orbsmax', 'st_mass']]` 进行处理

```
import pandas as pd
```

```
# 加载数据
```

```
df = pd.read_csv("D:/lianshidaiRecurit/03/PSCompPars_2025.02.09_07.27.02.csv")
```

```
# 处理缺失值
```

```
df = df.dropna(subset=['pl_orbsmax', 'st_mass', 'pl_orbper'])
```

```
# 选择相关特征和目标变量
```

```
X = df[['pl_orbsmax', 'st_mass']]
```

```
y = df['pl_orbper']
```

训练模型

首先需要导入必要的库

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

对于一个已经处理好的数据 x, y , 为了训练模型, 需要将 x, y 分别拆分成 **训练集** 和 **测试集** , 这一步可以使用 `train_test_split` 来随机拆分数数据集:

80% 用于训练 ($X_{\text{train}}, y_{\text{train}}$)

20% 用于测试 ($X_{\text{test}}, y_{\text{test}}$)

这一步可以用以下代码实现:

```
# 拆分数据集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

随后需要 **创建并训练模型** , 具体的实现方式是:

```
model = LinearRegression() # 创建线性回归模型
model.fit(X_train, y_train) # 训练模型 (让模型学习 X_train 和 y_train 之间的关系)
```

其中 `.fit(X_train, y_train)` 让模型学习数据的模式, 即找到最合适的 **直线** 来拟合数据。

接下来使用训练好的模型预测测试集

```
y_pred = model.predict(X_test) # 用训练好的模型预测测试集数据
```

使用 MSE 评估模型的预测性能

什么是 **MSE** ?

MSE指标是什么 ¹ ² ³

均方误差 (MSE, Mean Squared Error) 是一种常用的统计度量，用于评估预测模型或估计方法的精确性。它通过计算预测值与实际值之间差值的平方的平均值来衡量误差。MSE的值越小，说明模型的预测精度越高 ¹。

MSE的计算公式

MSE的计算公式如下：

$$MSE = (1/n) * \sum (y_i - y_{pred})^2$$

其中，n是样本数量， y_i 是实际值， y_{pred} 是预测值 ²。

MSE的特点

- 非负值：**MSE的值始终非负，最小值为0，这发生在所有的预测值都精确匹配实际值时。
- 惩罚较大误差：**由于误差项是平方的，所以较大的误差对MSE的贡献也更大，这意味着MSE对较大误差更为敏感。
- 量纲：**MSE的单位是原始数据单位的平方。例如，如果数据单位是米，则MSE的单位是平方米。
- 对异常值敏感：**由于误差平方的影响，MSE对异常值或离群值非常敏感 ³。

也就是说，**MSE** 反映了模型预测的准确程度，和准确率成正比

使用 python 进行实现如下：

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_test, y_pred)
print(f"均方误差 (MSE): {mse}")
```

绘图

如果需要将数据绘制成图表，需要 matplotlib 进行操作

其中

`plt.scatter()` 是 Matplotlib 用于绘制散点图的函数，它可以接受多个参数来控制散点的颜色、大小、形状等。

```
plt.scatter(x, y, s=None, c=None, marker=None, alpha=None, label=None, cmap=None, edgecolors=Nor
```

其中参数的含义为:

参数	说明
x	x 轴数据（横坐标）
y	y 轴数据（纵坐标）。
s	散点的大小，可以是单个数值或数组，默认 20。
c	颜色，可以是单个颜色值（如 'red'）或数值数组（结合 cmap 使用）。
marker	散点的形状，如 'o'（圆）、's'（方形）、'D'（菱形）等。
alpha	透明度，范围 [0,1]，0 为完全透明，1 为不透明。
label	用于图例的标签，配合 plt.legend() 使用。
cmap	颜色映射（当 c 为数值数组时），如 'viridis'、'plasma'、'coolwarm' 等。
edgecolors	散点的边缘颜色，如 'black'。
linewidths	边缘线宽度，默认为 None。

plt.plot() 是 Matplotlib 中用于绘制折线图或点线图的函数，它的参数可以控制线条样式、颜色、标记形状等。

plt.plot(x, y, fmt, color=None, linestyle=None, linewidth=None, marker=None, markersize=None, a:

参数	说明
x	x 轴数据（横坐标）。
y	y 轴数据（纵坐标）。
fmt	格式字符串，用于同时指定颜色、线型和标记（可选）。
color	颜色，如 'r'（红）、'g'（绿）、'b'（蓝）、'#ff5733'（十六进制）、(0.5,0.2,0.8)（RGB）。
linestyle	线型，如 '-'（实线）、'--'（虚线）、':'（点线）、'-.'（点划线）、'none'（无线）。
linewidth	线宽，默认 1.5。
marker	数据点的标记，如 'o'（圆）、's'（方形）、'D'（菱形）、'x'（叉号）、'v'（下三角）。

参数	说明
markersize	标记的大小。
alpha	透明度，取值 [0,1]，0 完全透明，1 不透明。
label	图例名称（需配合 plt.legend() 使用）。

```
import matplotlib.pyplot as plt

# 可视化预测结果与真实值的关系，设置不透明度为 0.5 以便观察点的重合情况
plt.scatter(y_test, y_pred, alpha=0.5)
# 绘制对角线
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
# 设置x轴、y轴以及标题名称
plt.xlabel("True pl_orbper")
plt.ylabel("Predicted pl_orbper")
plt.title("prediction vs True value")
# 绘制图像
plt.show()
```

导出线性方程参数

在 sklearn.linear_model.LinearRegression 中，训练好模型后，可以通过 .coef_ 和 .intercept_ 来获取最佳的 **权重** 和 **偏置**。最终的线性回归方程的形式如下：

$$y = w_1 \cdot x_1 + w_2 \cdot x_2 \cdots w_n \cdot x_n + b$$

其中：
 w_1, w_2, \dots, w_n 是特征的 **权重**，可通过 model.coef_ 获取。
 b 是 **偏置**，可通过 model.intercept_ 获取。

手动使用最小二乘法完成线性回归

这里使用 **正规方程** 进行计算最小二乘法

$$\theta = (X^T X)^{-1} X^T Y$$

详解正规方程 (Normal Equation)



Chenllliang

科学技术是第一生产力 喜欢《球状闪电》和《全频带阻塞干扰》

+ 关注他

366 人赞同了该文章

第一篇机器学习文章希望做到详细而优雅

相信学过[线性回归](#)的小伙伴对标题图片中的[方程式](#)一定不陌生。用的时候可能并不知其所以然，大一下学期我在学校学习完了线性代数和[多元函数微积分](#)的知识后，重新认识了这个方程

。（所有方程使用TeX重新编辑，带给你鸡汁的视觉体验！）

预备知识：[线性代数](#)，多元函数微积分（大一知识即可）

使用伪逆公式计算最小二乘法的参数

```
theta_best = np.linalg.pinv(X_b.T @ X_b) @ X_b.T @ y_train
```

这里得到的 `theta_best` 所包含的数据 $\theta_0, \theta_1, \theta_2 \cdots \theta_n$ 即最好的参数：

$$\hat{y} = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 \cdots \theta_n \cdot x_n$$

进行对数变化

python 中的 `numpy` 为对数变换提供了一个有效的工具，只需要调用 `np.log()` 就能做到对目标数据的对数变换操作，并返回相同数据格式的变换结果

进行以 10 为底的对数变换

```
X_log10 = np.log10(X)
```

输出R检测法结果

这里需要调用 `sklearn.metrics` 中的 `r2_score` 进行计算

```
from sklearn.metrics import r2_score
```

在操作上，只需要提供真实值和预测值，便能返回其线性回归的 R^2

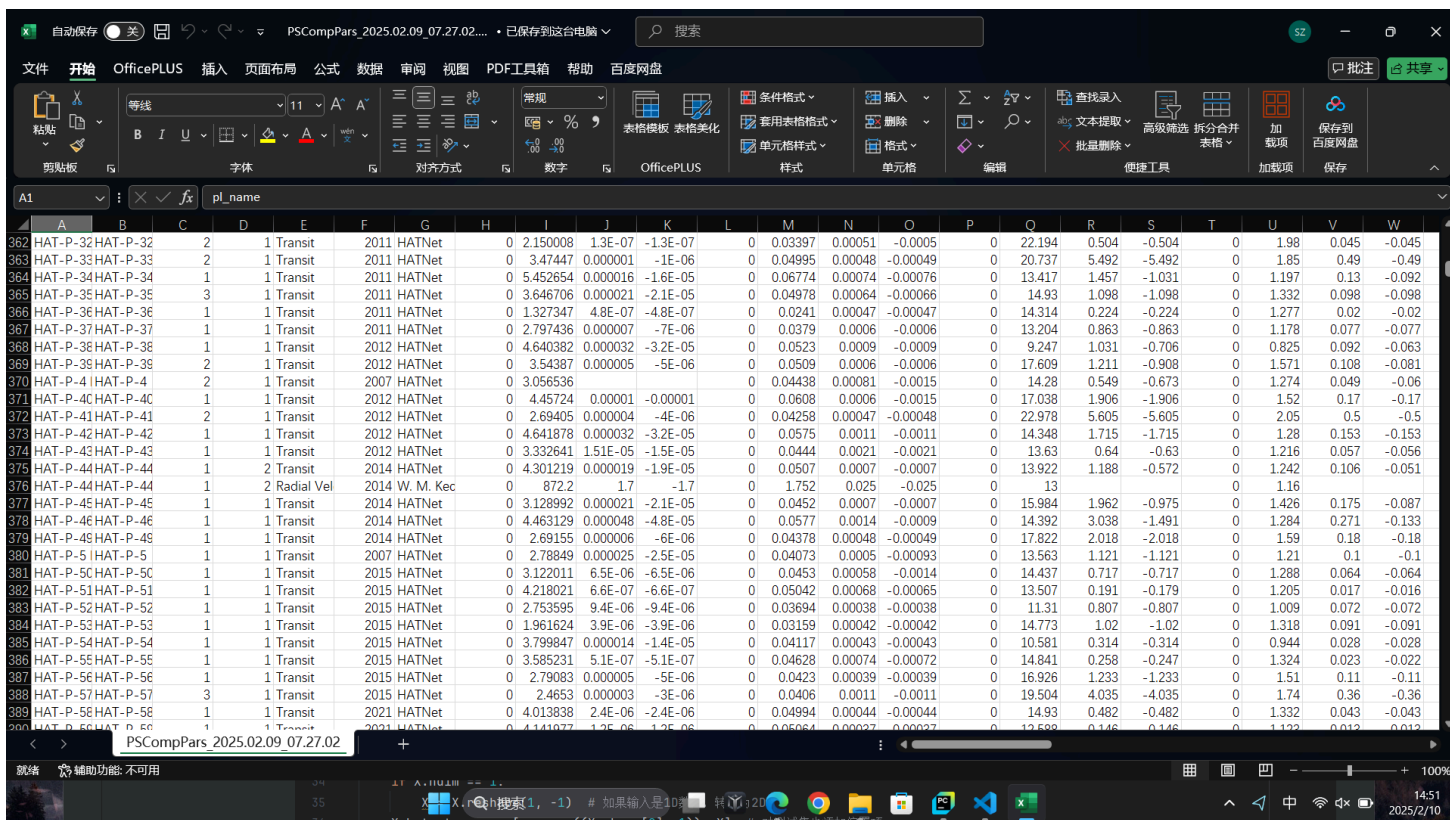
```
r2_sklearn = r2_score(y_test, y_pred)
```


流程记录

导入必要的库

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

第一步：访问 NASA 系外行星数据库，获取行星数据保存在 csv 文件中



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
362	HAT-P-32	HAT-P-32	2	1	Transit	2011	HATNet	0	2.150008	1.3E-07	-1.3E-07	0	0.03397	0.00051	-0.0005	0	22.194	0.504	-0.504	0	1.98	0.045	-0.045
363	HAT-P-33	HAT-P-33	2	1	Transit	2011	HATNet	0	3.47447	0.000001	-1E-06	0	0.04995	0.00048	-0.00049	0	20.737	5.492	-5.492	0	1.85	0.49	-0.49
364	HAT-P-34	HAT-P-34	1	1	Transit	2011	HATNet	0	5.452654	0.000016	-1.6E-05	0	0.06774	0.00074	-0.00076	0	13.417	1.457	-1.031	0	1.197	0.13	-0.092
365	HAT-P-35	HAT-P-35	3	1	Transit	2011	HATNet	0	3.646706	0.000021	-2.1E-05	0	0.04978	0.00064	-0.00066	0	14.93	1.098	-1.098	0	1.332	0.098	-0.098
366	HAT-P-36	HAT-P-36	1	1	Transit	2011	HATNet	0	1.327347	4.8E-07	-4.8E-07	0	0.0241	0.00047	-0.00047	0	14.314	0.224	-0.224	0	1.277	0.02	-0.02
367	HAT-P-37	HAT-P-37	1	1	Transit	2011	HATNet	0	2.797436	0.000007	-7E-06	0	0.0379	0.0006	-0.0006	0	13.204	0.863	-0.863	0	1.178	0.077	-0.077
368	HAT-P-38	HAT-P-38	1	1	Transit	2012	HATNet	0	4.640382	0.000032	-3.2E-05	0	0.0523	0.0009	-0.0009	0	9.247	1.031	-0.706	0	0.825	0.092	-0.063
369	HAT-P-39	HAT-P-39	2	1	Transit	2012	HATNet	0	3.54387	0.000005	-5E-06	0	0.0509	0.0006	-0.0006	0	17.609	1.211	-0.908	0	1.571	0.108	-0.081
370	HAT-P-4	HAT-P-4	2	1	Transit	2007	HATNet	0	3.056536			0	0.04438	0.00081	-0.0015	0	14.28	0.549	-0.673	0	1.274	0.049	-0.06
371	HAT-P-40	HAT-P-40	1	1	Transit	2012	HATNet	0	4.45724	0.00001	-0.00001	0	0.0608	0.0006	-0.0015	0	17.038	1.906	-1.906	0	1.52	0.17	-0.17
372	HAT-P-41	HAT-P-41	2	1	Transit	2012	HATNet	0	2.69405	0.000004	-4E-06	0	0.04258	0.00047	-0.00048	0	22.978	5.605	-5.605	0	2.05	0.5	-0.5
373	HAT-P-42	HAT-P-42	1	1	Transit	2012	HATNet	0	4.641878	0.000032	-3.2E-05	0	0.0575	0.0011	-0.0011	0	14.348	1.715	-1.715	0	1.28	0.153	-0.153
374	HAT-P-43	HAT-P-43	1	1	Transit	2012	HATNet	0	3.332641	1.51E-05	-1.5E-05	0	0.0444	0.0021	-0.0021	0	13.63	0.64	-0.63	0	1.216	0.057	-0.056
375	HAT-P-44	HAT-P-44	1	2	Transit	2014	HATNet	0	4.301219	0.000019	-1.9E-05	0	0.0507	0.0007	-0.0007	0	13.922	1.188	-0.572	0	1.242	0.106	-0.051
376	HAT-P-44	HAT-P-44	1	2	Radial Vel	2014	W. M. Kec	0	872.2	1.7	-1.7	0	1.752	0.025	-0.025	0	13			0	1.16		
377	HAT-P-45	HAT-P-45	1	1	Transit	2014	HATNet	0	3.128992	0.000021	-2.1E-05	0	0.0452	0.0007	-0.0007	0	15.984	1.962	-0.975	0	1.426	0.175	-0.087
378	HAT-P-46	HAT-P-46	1	1	Transit	2014	HATNet	0	4.463129	0.000048	-4.8E-05	0	0.0577	0.0014	-0.0009	0	14.392	3.038	-1.491	0	1.284	0.271	-0.133
379	HAT-P-48	HAT-P-48	1	1	Transit	2014	HATNet	0	2.69155	0.000006	-6E-06	0	0.04378	0.00048	-0.00049	0	17.822	2.018	-2.018	0	1.59	0.18	-0.18
380	HAT-P-5	HAT-P-5	1	1	Transit	2007	HATNet	0	2.78849	0.000025	-2.5E-05	0	0.04073	0.0005	-0.00093	0	13.563	1.121	-1.121	0	1.21	0.1	-0.1
381	HAT-P-50	HAT-P-50	1	1	Transit	2015	HATNet	0	3.122011	6.5E-06	-6.5E-06	0	0.0453	0.00058	-0.0014	0	14.437	0.717	-0.717	0	1.288	0.064	-0.064
382	HAT-P-51	HAT-P-51	1	1	Transit	2015	HATNet	0	4.218021	6.6E-07	-6.6E-07	0	0.05042	0.00068	-0.00065	0	13.507	0.191	-0.179	0	1.205	0.017	-0.016
383	HAT-P-52	HAT-P-52	1	1	Transit	2015	HATNet	0	2.753595	9.4E-06	-9.4E-06	0	0.03694	0.00038	-0.00038	0	11.31	0.807	-0.807	0	1.009	0.072	-0.072
384	HAT-P-53	HAT-P-53	1	1	Transit	2015	HATNet	0	1.961624	3.9E-06	-3.9E-06	0	0.03159	0.00042	-0.00042	0	14.773	1.02	-1.02	0	1.318	0.091	-0.091
385	HAT-P-54	HAT-P-54	1	1	Transit	2015	HATNet	0	3.799847	0.000014	-1.4E-05	0	0.04117	0.00043	-0.00043	0	10.581	0.314	-0.314	0	0.944	0.028	-0.028
386	HAT-P-55	HAT-P-55	1	1	Transit	2015	HATNet	0	3.585231	5.1E-07	-5.1E-07	0	0.04628	0.00074	-0.00072	0	14.841	0.258	-0.247	0	1.324	0.023	-0.022
387	HAT-P-56	HAT-P-56	1	1	Transit	2015	HATNet	0	2.79093	0.000005	-5E-06	0	0.0423	0.00039	-0.00039	0	16.926	1.233	-1.233	0	1.51	0.11	-0.11
388	HAT-P-57	HAT-P-57	3	1	Transit	2015	HATNet	0	2.4653	0.000003	-3E-06	0	0.0406	0.0011	-0.0011	0	19.504	4.035	-4.035	0	1.74	0.36	-0.36
389	HAT-P-58	HAT-P-58	1	1	Transit	2021	HATNet	0	4.013838	2.4E-06	-2.4E-06	0	0.04994	0.00044	-0.00044	0	14.93	0.482	-0.482	0	1.332	0.043	-0.043
390	HAT-P-59	HAT-P-59	1	1	Transit	2021	HATNet	0	1.141037	1.0E-06	-1.0E-06	0	0.05651	0.00037	-0.00037	0	12.562	0.145	-0.145	0	1.122	0.012	-0.012

第二步：提取 csv 文件中所需要的数据，保留'pl_orbsmax', 'st_mass'和'pl_orbper'数据，与此同时注意到某些数据是缺失的，故而使用 df.dropna(subset=['pl_orbsmax', 'st_mass', 'pl_orbper']) 用来删除含有缺失值的数据组，保证后续计算正确

```
# 加载数据
df = pd.read_csv("D:/lianshidaiRecurit/03/PSCompPars_2025.02.09_07.27.02.csv")

# 处理缺失值
df = df.dropna(subset=['pl_orbsmax', 'st_mass', 'pl_orbper'])

# 选择相关特征和目标变量
X = df[['pl_orbsmax', 'st_mass']]
y = df['pl_orbper']
```

第三步：使用提取到的数据来训练模型

首先需要将数据使用 `train_test_split` 来拆分成 8: 2 的训练集和测试集，使用训练集来进行线性回归拟合

```
# 拆分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 使用 sklearn 进行线性回归
model = LinearRegression()
model.fit(X_train, y_train)

# 预测
y_pred = model.predict(X_test)
```

第四步：使用 MSE、R检测法 来评估模型的预测性能

```
# 预测
y_pred = model.predict(X_test)

# 计算 sklearn 线性回归的 R^2 和 MSE
r2_sklearn = r2_score(y_test, y_pred)
mse_sklearn = mean_squared_error(y_test, y_pred)
```

第五步：手动使用最小二乘法来完成线性回归的任务，这里具体的计算方法使用了 伪逆矩阵

```

def manual(X_train, X_test, y_train, y_test):
    """手动实现最小二乘法计算参数"""
    # 偏置项
    X_b = np.c_[np.ones((len(X_train), 1)), X_train.values]
    # 伪逆法求解 theta_best
    theta_best = np.linalg.pinv(X_b.T @ X_b) @ X_b.T @ y_train

    # 手动预测函数
    def predict_manual(X):
        X = X.values # 转换为 NumPy 数组
        X_b_test = np.c_[np.ones((X.shape[0], 1)), X]
        return X_b_test @ theta_best

    # 计算手动最小二乘法的预测值
    y_manual_pred = predict_manual(X_test)

    # 计算手动实现的 R^2 和 MSE
    r2_manual = r2_score(y_test, y_manual_pred)
    mse_manual = mean_squared_error(y_test, y_manual_pred)

    return r2_manual, mse_manual, theta_best, y_manual_pred

```

第六步： 将预测值的真实值的关系使用 matplotlib 绘制散点图进行 **可视化**

```

def draw_plot_map():
    # 可视化预测结果
    plt.scatter(y_test, y_pred, label="Sklearn Predict", alpha=0.5)
    plt.scatter(y_test, y_manual_pred, label="Manual Least Squares Predict", alpha=0.5, marker='x')
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
    plt.xlabel("True orbital period")
    plt.ylabel("Predicted orbital period")
    plt.title("Prediction vs True value (Log Transformed)")
    plt.legend()
    plt.show()

```

第七步： 输出最终的线性方程参数

```

def output():
    # 输出结果
    print("Sklern 线性回归模型: ")
    print(f"权重: {intercept_sklern}")
    print(f"偏置: {coef_sklern}")
    print(f"R²: {r2_sklern}")
    print("log10(MSE):", log_mse_sklern)
    print(f"Sklern: y = {model.intercept_:.4f} + {model.coef_[0]:.4f} * p1_orbsmax + {model.coef_[1]:.4f} * p2_orbsmax")
    print("-----")
    print("手动最小二乘法模型: ")
    print(f"计算出的 theta_best: {theta_best.flatten()}")
    print(f"R²: {r2_manual}")
    print("log10(MSE):", log_mse_manual)
    print(f"Manual: y = {theta_best[0]:.4f} + {theta_best[1]:.4f} * p1_orbsmax + {theta_best[2]:.4f} * p2_orbsmax")

```

第八步： 对数据进行对数变换，然后重复第二到七步，获得模型效果，进行对比

在 **天体问题** 中，习惯的对数变换是以 **10为底数** 的变换，更符合天文物理学的习惯。与此同时，取对数可以更直观地对应开普勒定律。

这里需要时刻注意何时使用 **对数后的数据** 何时使用 **原数据**

具体使用代码来实现如下：

```

def pre():
    """训练前的数据处理"""

    #其余部分不变，略去

    # 进行 log10 对数变换
    X_log10 = np.log10(X)
    y_log10 = np.log10(y)

    # 拆分训练集和测试集
    X_train, X_test, y_train, y_test = train_test_split(X_log10, y_log10, test_size=0.2, random_

    return X_train, X_test, y_train, y_test

def sklearn(X_train, X_test, y_train, y_test):
    # 使用 sklearn 进行线性回归
    model = LinearRegression()
    model.fit(X_train, y_train)

    # 预测
    y_pred_log = model.predict(X_test)

    # 反变换回原始尺度
    y_pred = 10 ** y_pred_log

    # 计算 sklearn 线性回归的 R^2 和 MSE,注意在 log10 尺度计算 R^2 , MSE 需要在原尺度计算
    r2_sklearn = r2_score(y_test, y_pred_log)
    mse_sklearn = mean_squared_error(10 ** y_test, y_pred)

    return r2_sklearn, mse_sklearn, model, y_pred

```

第九步：解释模型的误差

各个系数的物理含义：

回归方程：

$$\log_{10} pl_{orbper} = 2.5611 + 1.4962 \log_{10} pl_{orbsmax} - 0.4700 \log_{10} st_{mass}$$

对比理论的开普勒第三定律：

$$pl_{orbper} = C \cdot pl_{orbsmax}^{3/2} \cdot st_{mass}^{-1/2}$$

取对数：

$$\log_{10} pl_orbper = \log_{10} C + 1.5 \log_{10} pl_orbsmax - 0.5 \log_{10} st_mass$$

其中，**偏置项** 代表：

$$\log_{10} C$$

所以，我们可以反推：

$$C = 10^{2.5611} \approx 363$$

开普勒定律的系数 c 取决于单位, 在以 **AU、天和太阳质量** 为单位的时候，理论值为365.25
而由公式可得，另两个参数的理论值应该分别为 **1.5** 和 **-0.5**

最终的参数 **实际值** 和 **理论值** 对比：

$$363 \approx 365.25$$

$$1.4962 \approx 1.5$$

$$-0.4700 \approx -0.5$$

尽管各个系数非常接近 **理论值**，但不完全一致，可能有几个原因：

1. 数据误差

- 测量数据可能有误差，导致拟合出的参数略有偏差

2. 情况复杂

- 理论上行星轨道只受到半径和质量的影响，但是在现实情况下，会受到其他天体（附近的行星）的引力扰动，这使得产生了噪声

如果希望修复这问题，我有以下两个思路：

1. 增大数据量

- 通过增大训练的数据量，可以一定程度上减少测量带来的误差影响

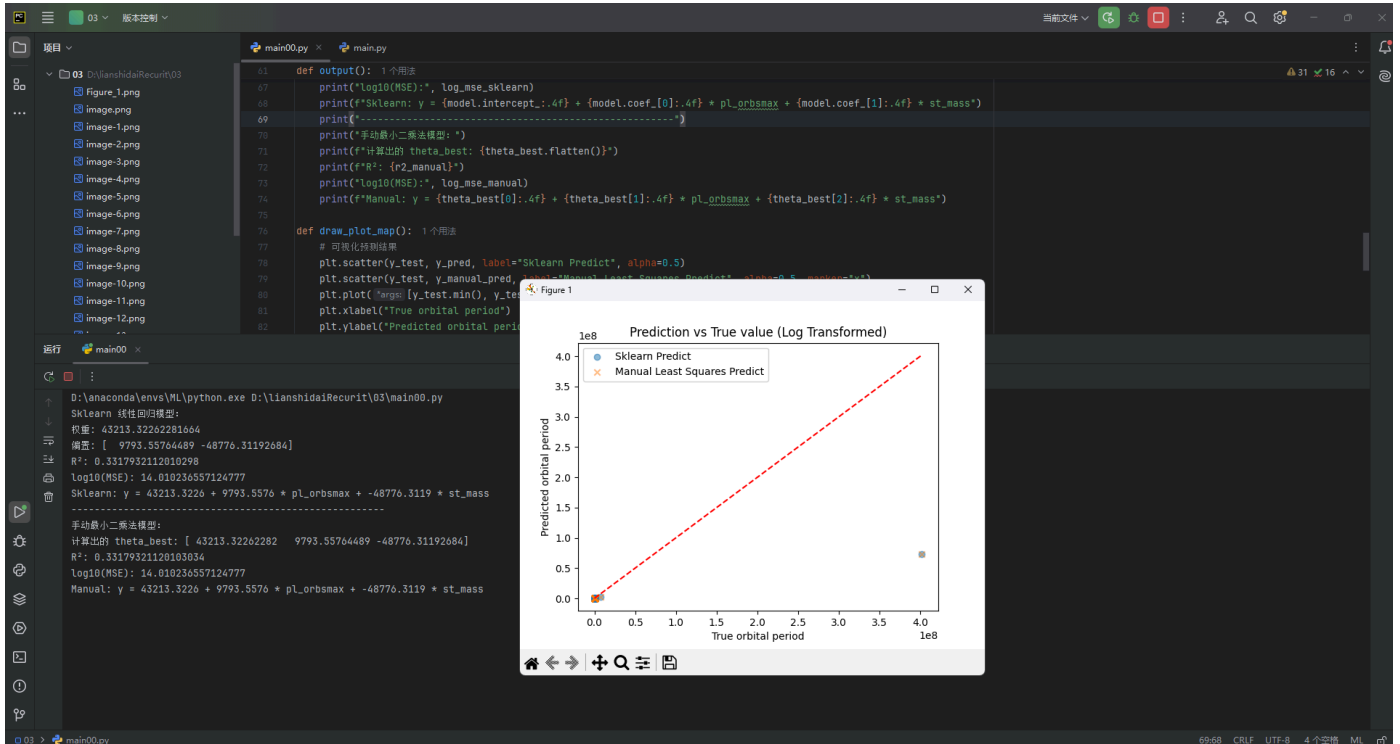
2. 删除噪声过大的数据

- 对于某些行星而言，其运动周期受到其他天体影响过于巨大，对于这样的天体再纳入训练只会起到反面作用。

- 可以通过计算数据的 **残差**，然后删除掉残差超过设定的 **阈值** 的数据，即噪声过大的数据来实现更精准模型

运行成功截图

- 对数变化前：



- 对数变化（10为底数）后：

