Машинне навчання з scikit-learn

Загалом, більшість об’єктів scikit-learn мають наступні методи, залежно від того, для чого вони використовуються:

fit()	Тренує модель або перетворювач
transform()	Перетворює дані
fit_transform()	Запуск fit(), потім transform()
score()	Оцінка моделі
predict()	Прогнозування
fit_predict()	Запуск fit(), потім predict()
predict_proba()	Прогнозування, що повертає ймовірність належності до класу

Машинне навчання з scikit-learn

linear_model містить моделі лінійної регресії. Існує проста лінійна регресія, що має одну незалежну змінну та множинна лінійна регресія, де незалежних змінних кілька.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

Лінійна регресія в scikit-learn використовує звичайні найменші квадрати (OLS), що дає коефіцієнти, які мінімізують суму квадратів похибок (виміряних як відстань між y та y’). Коефіцієнти можна знайти за допомогою рішення закритої форми або оцінити за допомогою методів оптимізації, таких як градієнтний спуск, який використовує негативний градієнт (напрям найбільш крутого підйому, розрахований за допомогою часткових похідних), щоб визначити, які коефіцієнти слід спробувати далі.

Машинне навчання з scikit-learn

Наприклад, потрібно передбачити довжину періоду за масою, ексцентриситетом та великою піввісю.

Спочатку потрібно виділити з даних незалежні змінні та залежну.

```
data = planet_data[['semimajoraxis', 'period', 'mass', 'eccentricity']].dropna()
X = data[['semimajoraxis', 'mass', 'eccentricity']]
y = data.period
```

Потім розділити дані на навчальний та тестовий набори:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

Машинне навчання з scikit-learn

І створити модель:

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression().fit(X_train, y_train)
```

```
lm.intercept_
-622.9909910671802
```

```
lm.coef_
array([ 1880.43659904,   -90.18675917, -3201.07805933])
```

Машинне навчання з scikit-learn

Після створення моделі можна використовувати її для прогнозування. Але спочатку треба перевірити її роботу на тестовому наборі.

```
preds = lm.predict(X_test)
```

```
np.corrcoef(y_test, preds)[0][1]
```

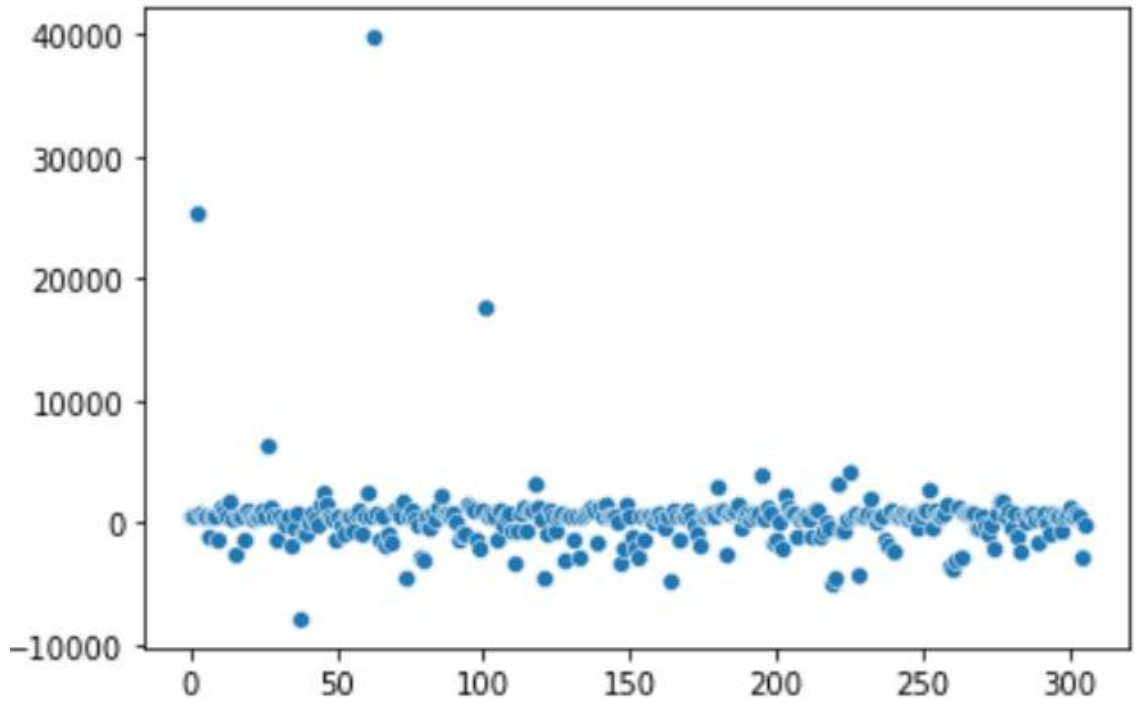
0.9692104355988059

Машинне навчання з scikit-learn

Щоб оцінити модель лінійної регресії, потрібно візуалізувати залишки (розбіжності між фактичними значеннями та прогнозами моделі); вони повинні бути зосереджені навколо нуля і мати однакову дисперсію. Можна використати ядерну оцінку щільності, щоб оцінити, чи залишки зосереджені навколо нуля, і діаграму розсіювання, щоб побачити, чи мають вони однакову дисперсію.

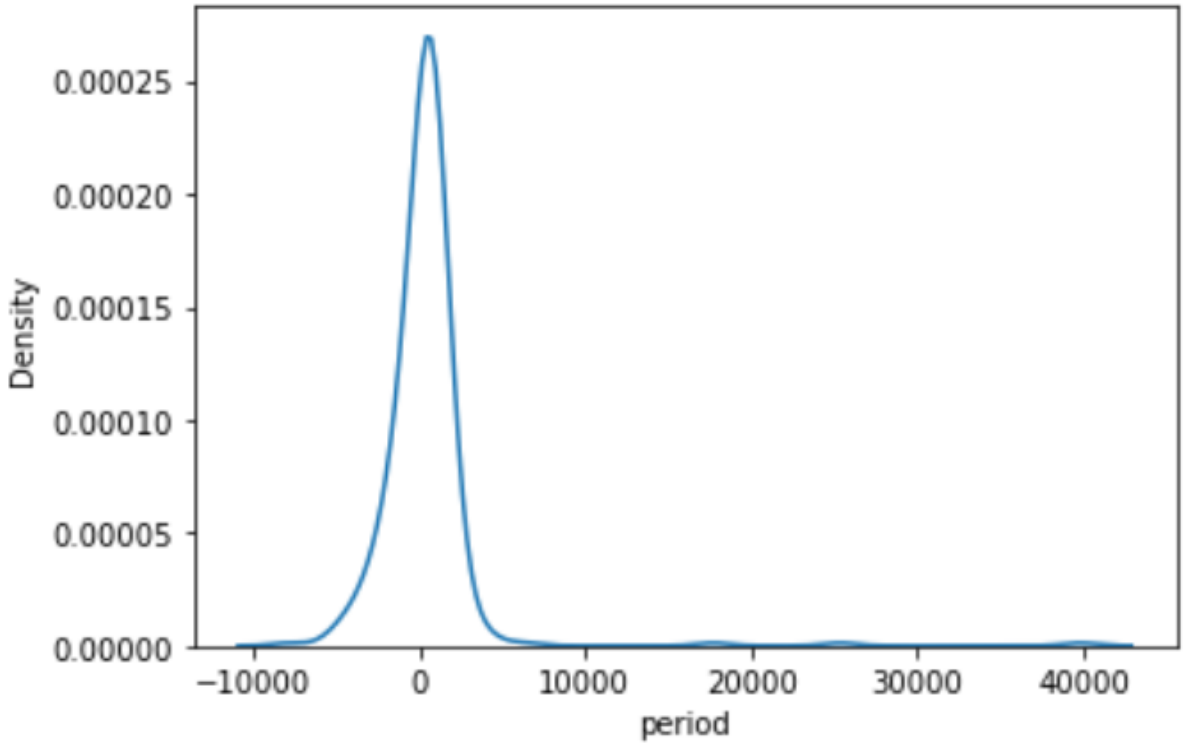
Машинне навчання з scikit-learn

```
residuals = y_test-preds
sns.scatterplot(np.arange(residuals.shape[0]), residuals)
```



Машинне навчання з scikit-learn

sns.kdeplot(residuals)



Машинне навчання з scikit-learn

Метод score() лінійної регресійної моделі повертає параметр R^2, або коефіцієнт детермінації, який кількісно визначає частку дисперсії залежної змінної, яку можна передбачити на основі незалежних змінних. Цей коефіцієнт має значення від 0 до 1, і чим ближчий він до одиниці, тим краще.

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

```
lm.score(X_test, y_test)
```
0.9209013475842683
```
from sklearn.metrics import r2_score
r2_score(y_test, preds)
```
0.9209013475842683

Машинне навчання з scikit-learn

Ще одна метрика, яку пропонує scikit-learn, — це оцінка поясненої дисперсії, яка говорить нам про відсоток дисперсії, що пояснюється моделлю. Потрібно, щоб це значення було якомога ближче до 1:

```
from sklearn.metrics import explained_variance_score
explained_variance_score(y_test, preds)
```

0.922014421842937 1

Середня абсолютна похибка (MAE) показує середню похибку моделі в будь-якому напрямку. Значення коливаються від 0 до ∞ (нескінченність), причому менші значення є кращими:

$$MAE = \frac{\sum_i |y_i - \hat{y}_i|}{n}$$

Машинне навчання з scikit-learn

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, preds)
```

1369.4418170735335

Корінь середньоквадратичної помилки (RMSE) дозволяє додатково оцінити прогнозування:

$$RMSE = \sqrt{\frac{\sum_i (y_i - \hat{y}_i)^2}{n}}$$

```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test, preds))
```

3248.499961928377

Машинне навчання з scikit-learn

Альтернативою всім цим вимірюванням на основі середнього значення є середня абсолютна помилка, яка є медіаною залишків. Її можна використовувати в тих випадках, коли є кілька викидів у залишках, і потрібно отримати більш точной опис основної частини похибок.

```
from sklearn.metrics import median_absolute_error
median_absolute_error(y_test, preds)
```

```
759.8613358335447
```

Машинне навчання з scikit-learn

Кластеризація використовується, щоб розділити точки даних на групи подібних точок. Точки в кожній групі більше схожі на точки зі своєї групи, ніж з інших груп.

Популярний алгоритм для кластеризації – це алгоритм k-середніх, який ітераційно призначає точки найближчій групі, використовуючи відстань від центру групи, створюючи k груп. Оскільки в цій моделі використовуються обчислення відстані, необхідно заздалегідь зрозуміти, який вплив на результати матиме шкала.

Машинне навчання з scikit-learn

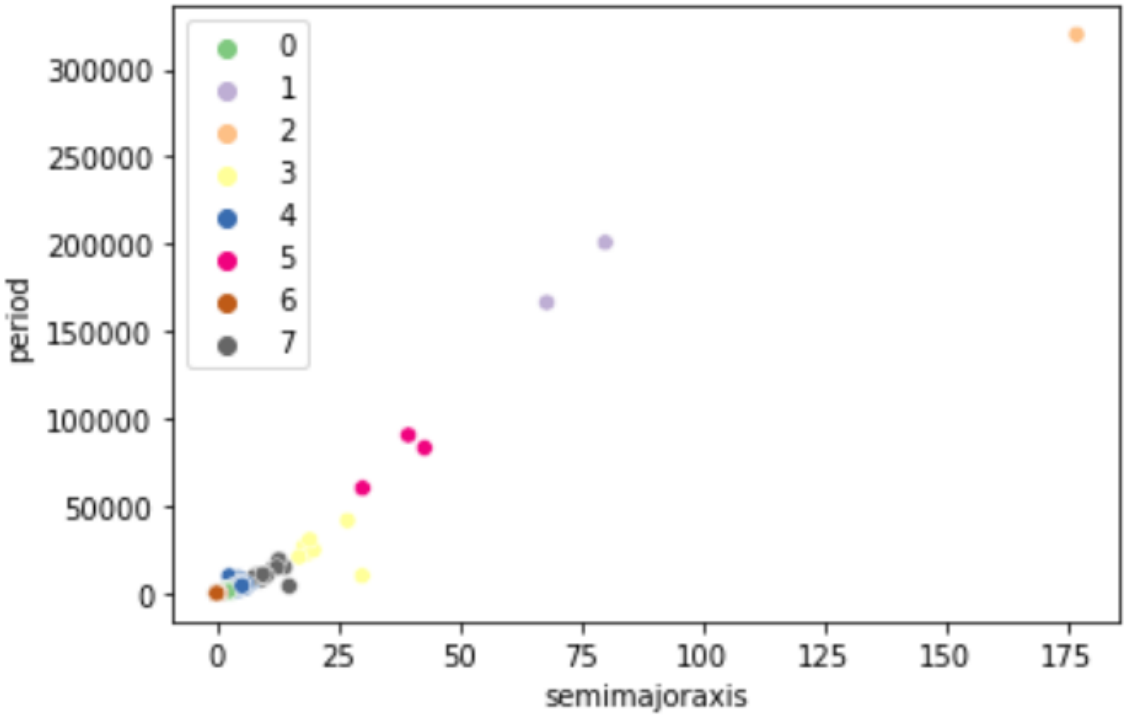
Оскільки алгоритм обрає початкові центри випадково, то можна отримати різний результат при різних запусках. Щоб уникнути цього, можна вставити значення параметру random_state.

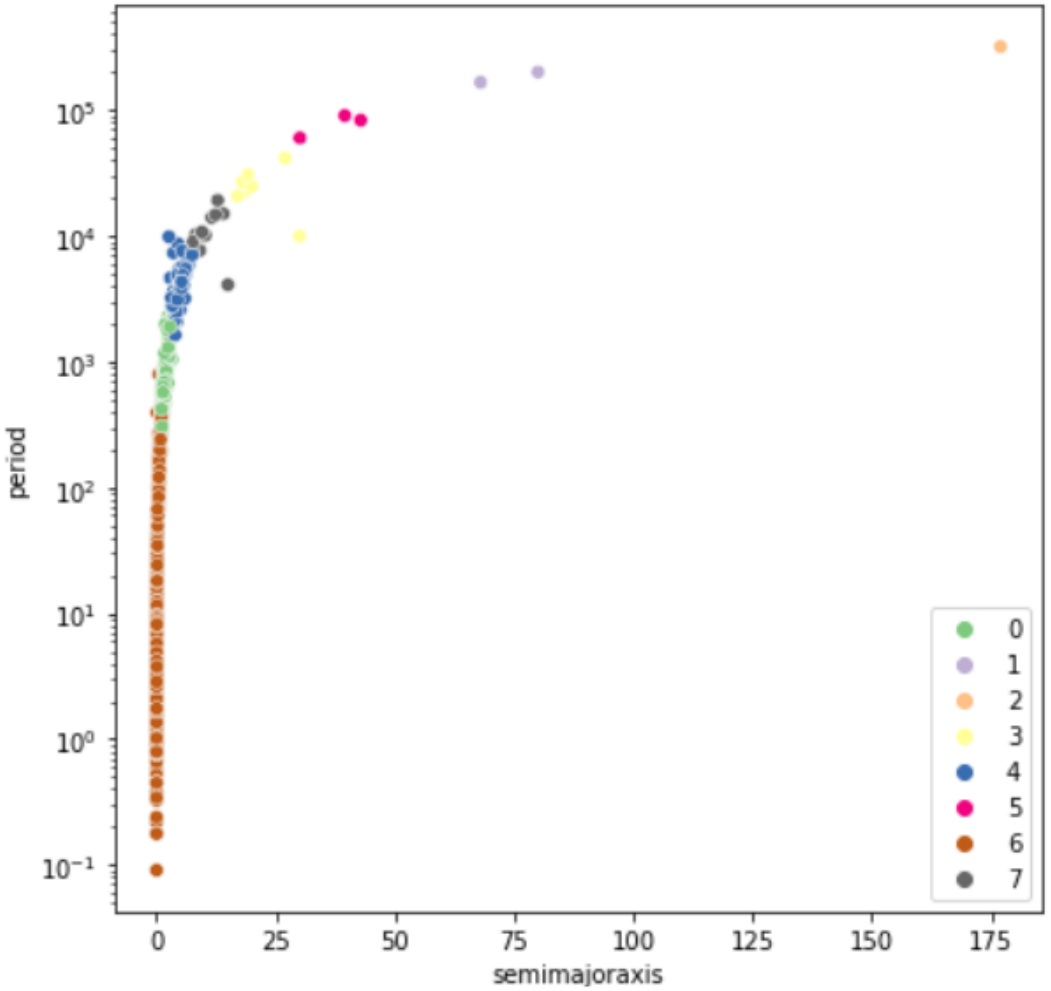
```
from sklearn.cluster import KMeans
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
kmeans_pipeline = Pipeline([('scale', StandardScaler()),('kmeans', KMeans(8, random_state=0))])
kmeans_data = planet_data[['semimajoraxis', 'period']].dropna()
kmeans_pipeline.fit(kmeans_data)
```

```
Pipeline(steps=[('scale', StandardScaler()),
                ('kmeans', KMeans(random_state=0))])
```

Машинне навчання з scikit-learn

sns.scatterplot(x=kmeans_data.semimajoraxis,y=kmeans_data.period, hue=kmeans_pipeline.predict(kmeans_data),palette='Accent’)





Машинне навчання з scikit-learn

Для того, щоб оцінити виконання задачі кластеризації у випадку навчання без учителя, потрібно використовувати показники, які оцінюють аспекти самих кластерів, наприклад, наскільки вони віддалені один від одного і наскільки близько розташовані точки в кластері. Можна порівняти кілька показників, щоб отримати більш повну оцінку ефективності.

Один з таких методів називається коефіцієнтом силуету, який допомагає кількісно визначити поділ кластерів. Він обчислюється шляхом віднімання середнього значення відстаней між кожними двома точками в кластері (a) із середнього відстаней між точками в даному кластері та найближчого іншого кластера (b) і діленням на максимальну з двох:

$$\frac{b - a}{\max(a, b)}$$

Машинне навчання з scikit-learn

Ця метрика повертає значення в діапазоні [-1, 1], де -1 є найгіршим (кластери призначені неправильно), а 1 найкращим; значення поблизу 0 вказують, що кластери перекриваються. Чим вище це число, тим краще визначені (більш відокремлені) кластери:

```
from sklearn.metrics import silhouette_score
silhouette_score(kmeans_data,
kmeans_pipeline.predict(kmeans_data))
```

0.7575062308971886

Машинне навчання з scikit-learn

Ще одна оцінка, яку можна використовувати для оцінки результату кластеризації, — це відношення відстаней всередині кластера (відстаней між точками в кластері) до відстаней між кластерами (відстаней між точками в різних кластерах), яке називається оцінкою Девіса-Болдіна. Значення ближчі до нуля вказують на кращі розділи між кластерами:

```
from sklearn.metrics import davies_bouldin_score
davies_bouldin_score(kmeans_data,
kmeans_pipeline.predict(kmeans_data))
```

0.4630237968027361

Машинне навчання з scikit-learn

Ще одним показником є оцінка Калінського та Харабаса, або критерій співвідношення дисперсії, який є відношенням дисперсії всередині кластера до дисперсії між кластерами. Вищі значення вказують на краще визначені (більш відокремлені) кластери:

```
from sklearn.metrics import calinski_harabasz_score
calinski_harabasz_score(kmeans_data,
kmeans_pipeline.predict(kmeans_data))
```

21200.08560545858