Машинне навчання з scikit-learn

Машинне навчання — це підмножина штучного інтелекту (ШІ), за допомогою якого алгоритм може навчитися передбачати значення на основі вхідних даних без явного вивчення правил. Ці алгоритми покладаються на статистичні дані, щоб робити висновки під час навчання; потім вони використовують те, чого навчилися, щоб робити прогнози.

Машинне навчання з scikit-learn

Машинне навчання зазвичай поділяється на три категорії: навчання без учителя, навчання з учителем і навчання з підкріпленням.

Навчання без учителя використовується тоді, коли немає позначених даних, тобто для жодних даних невідомо, до якого класу вони належать.

Якщо є позначені дані, то використовується навчання з учителем.

Навчання з підкріпленням пов’язане з реакцією на зворотний зв’язок із оточення.

Машинне навчання з scikit-learn

Найпоширенішими завданнями машинного навчання є кластеризація, класифікація та регресія.

В задачах кластеризації метою є розподілення даних за групами, при цьому групи повинні бути чітко визначеними, тобто члени групи знаходяться близько один до одного, а самі групи - відокремлені від інших груп.

Класифікація має на меті присвоєння мітки класу даним. При цьому мітка є дискретною.

Регресія, з іншого боку, призначена для передбачення числових значень; вона моделює силу і величину зв'язків між змінними.

scikit-learn – бібліотека для машинного навчання на Python, найбільш проста у використанні.

Машинне навчання з scikit-learn

Розглянемо дані про планети:

```
RangeIndex: 4094 entries, 0 to 4093
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   mass             1659 non-null   float64
 1   description      3709 non-null   object
 2   periastrontime   203 non-null    float64
 3   semimajoraxis    1704 non-null   float64
 4   discoveryyear    4083 non-null   float64
 5   list             4094 non-null   object
 6   eccentricity     1388 non-null   float64
 7   period           3930 non-null   float64
 8   discoverymethod  4046 non-null   object
 9   lastupdate       4087 non-null   object
 10  periastron       561 non-null    float64
 11  name             4094 non-null   object
dtypes: float64(7), object(5)
```

	mass	periastrontime	semimajoraxis	discoveryyear	eccentricity	period	periastron
count	1659.000000	2.030000e+02	1704.000000	4083.000000	1388.000000	3930.000000	561.000000
mean	2.702061	2.541145e+06	5.837964	2013.939995	0.159016	524.084969	132.081840
std	8.526177	1.566478e+06	110.668743	6.006576	0.185041	7087.428665	122.343308
min	0.000008	2.452942e+05	0.004420	1781.000000	0.000000	0.090706	-233.000000
25%	0.085000	2.452454e+06	0.051575	2014.000000	0.013000	4.552475	45.000000
50%	0.830000	2.454114e+06	0.140900	2015.000000	0.100000	12.364638	118.258489
75%	2.440000	2.455426e+06	1.190000	2016.000000	0.230000	46.793136	227.005878
max	263.000000	2.453248e+07	3500.000000	2020.000000	0.956000	320000.000000	791.000000

Наприклад, потрібно передбачити період планети, виходячи з її маси, ексцентриситету та великої піввісі.

Машинне навчання з scikit-learn

Якщо передати моделі всі дані для навчання, є ризик ситуації «перенавчання» моделі, модель буде занадто складною і не зможе правильно передбачати значення даних в нових точках. З іншого боку, якщо не надати моделі достатньо даних, вона вийде занадто простою і не зможе адекватно виявити інформацію, закономірності в даних, а, значить, не зможе їх передбачати.

Машинне навчання з scikit-learn

Потрібно розділити дані на навчальний набір і набір для тестування. Для цього можна перетасувати датафрейм і вибрати верхні x% рядків для навчання, а решту залишити для тестування. Але scikit-learn надає функцію train_test_split() у модулі model_selection, яка є більш надійним і простим у використанні рішенням. Для її використання потрібно попередньо відокремити вхідні дані (X) від вихідних даних (y).

Зазвичай, обирають об’єм набору для тестування від 10% до 30% даних.

Машинне навчання з scikit-learn

sklearn.model_selection.train_test_split()

test_size – розмір тестової вибірки, задається кількість рядків або частка. Якщо не вказаний розмір навчальної вибірки, то значення за замовчуванням – 0.25

train_size – розмір навчальної вибірки

shuffle – чи перемішувати дані перед розбиттям, за замовчуванням True

```
from sklearn.model_selection import train_test_split
X = planet_data[['mass','eccentricity', 'semimajoraxis']]
y = planet_data.period
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

X_train.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3070 entries, 1390 to 2732
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   mass           1225 non-null   float64
 1   eccentricity   1019 non-null   float64
 2   semimajoraxis  1260 non-null   float64
dtypes: float64(3)
```

y_train

```
1390      1.434742
2837     51.079263
3619      7.171000
1867     51.111024
1869     62.869161
             ...
835     466.000000
3264      6.400876
1653      8.630433
2607     44.431014
2732     11.837549
Name: period, Length: 3070, dtype: float64
```

X_train.describe().T

	count	mean	std	min	25%	50%	75%	max
mass	1225.0	2.460026	5.148323	0.000008	0.085000	0.8036	2.40000	79.000
eccentricity	1019.0	0.155856	0.181519	0.000000	0.010850	0.0980	0.23000	0.956
semimajoraxis	1260.0	4.566550	82.746237	0.004420	0.051348	0.1328	1.16175	2880.000

Машинне навчання з scikit-learn

Попередня підготовка даних до використання в моделях машинного навчання включає:

- масштабування та центрування даних;
- кодування даних;
- визначення відсутніх даних та їх заповнення;
- інші перетворення даних.

Машинне навчання з scikit-learn

Scikit-learn має модуль preprocessing для операцій з перетворення даних.

Клас StandardScaler здійснює стандартизацію даних, тобто приводить їх до стандартного нормального розподілу.

```
from sklearn.preprocessing import StandardScaler
standardized = StandardScaler().fit_transform(X_train)
standardized[~np.isnan(standardized)][:15]
```

```
array([-0.05436182,  1.43278593, -0.19626563,  1.95196592,  0.00451498,
       -0.31285603,  0.07795916, -0.04747176, -0.2463995 , -0.41810145,
       -0.05475873,  1.65946487, -0.40457381, -0.85904421, -0.05475111])
```

Машинне навчання з scikit-learn

Клас MinMaxScaler здійснює нормалізацію даних, тобто перераховує всі значення в діапазон [0,1].

```
from sklearn.preprocessing import MinMaxScaler
normalized = MinMaxScaler().fit_transform(X_train)
normalized[~np.isnan(normalized)][:15]
```

```
array([2.28055906e-05, 1.24474091e-01, 1.83543340e-02, 5.33472803e-01,
       1.71374569e-03, 1.07593965e-02, 1.77824268e-01, 2.20687839e-04,
       1.50885109e-02, 8.36820084e-02, 1.14062675e-05, 1.39240422e-01,
       4.78471239e-03, 0.00000000e+00, 1.16250178e-05])
```

Клас RobustScaler використовує медіану та міжквартильний розмах для масштабування з врахуванням викидів.

Машинне навчання з scikit-learn

Кодування даних перетворює категоріальні змінні у числові. Для бінарного кодування використовується клас LabelBinarizer.

df1

	Family	Father	Mother	Gender	Height	Kids
0	1	78.5	67.0	M	73.2	4
1	1	78.5	67.0	F	69.2	4
2	1	78.5	67.0	F	69.0	4
3	1	78.5	67.0	F	69.0	4
4	2	75.5	66.5	M	73.5	4

```
from sklearn.preprocessing import LabelBinarizer
df1.Gender=LabelBinarizer().fit_transform(df1.Gender)
```

	Family	Father	Mother	Gender	Height	Kids
0	1	78.5	67.0	1	73.2	4
1	1	78.5	67.0	0	69.2	4
2	1	78.5	67.0	0	69.0	4
3	1	78.5	67.0	0	69.0	4
4	2	75.5	66.5	1	73.5	4

Машинне навчання з scikit-learn

Також можна створити змінні-індикатори, що є додатковими булевими змінними та показують приналежність до певної групи.

```
planet_data.list.value_counts()
```

```
Confirmed planets                    3972
Controversial                          97
Retracted planet candidate             11
Solar System                            9
Kepler Objects of Interest              4
Planets in binary systems, S-type       1
Name: list, dtype: int64
```

```
pd.get_dummies(planet_data.list).head()
```

	Confirmed planets	Controversial	Kepler Objects of Interest	Planets in binary systems, S-type	Retracted planet candidate	Solar System
0	1	0	0	0	0	0
1	1	0	0	0	0	0
2	1	0	0	0	0	0
3	1	0	0	0	0	0
4	0	1	0	0	0	0

Машинне навчання з scikit-learn

pd.get_dummies(planet_data.list, drop_first=True).head()

	Controversial	Kepler Objects of Interest	Planets in binary systems, S-type	Retracted planet candidate	Solar System
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	1	0	0	0	0

Машинне навчання з scikit-learn

Для перетворення категоріальної характеристики з кількома значеннями використовується клас LabelEncoder або OrdinalEncoder.

	Age	Gender	Region	Occupation	Income	Has Laptop
0	14	male	city	student	0	no
1	34	female	city	teacher	22000	no
2	42	male	countryside	banker	24000	yes
3	30	male	countryside	teacher	25000	no
4	16	male	city	student	0	no
5	33	female	city	banker	20000	yes

```
from sklearn.preprocessing import LabelEncoder
LabelEncoder().fit_transform(df.Occupation).T[:15]
```

```
array([2, 3, 0, 3, 2, 0, 2, 2, 3, 2, 2, 2, 0, 0, 2])
```

Машинне навчання з scikit-learn

Класи для роботи з відсутніми значеннями у scikit-learn знаходяться у модулі impute.

from sklearn.impute import MissingIndicator
MissingIndicator().fit_transform(planet_data[['semimajoraxis', 'mass', 'eccentricity']])

```
array([[False, False, False],
       [False, False, False],
       [False, False, False],
       ...,
       [ True, False, False],
       [ True, False, False],
       [ True, False, False]])
```

Машинне навчання з scikit-learn

SimpleImputer за замовчуванням заповнює середніми значеннями. Можна змінити параметр strategy на інше значення: median, most_frequent або constant

from sklearn.impute import SimpleImputer
SimpleImputer().fit_transform(planet_data[['semimajoraxis', 'mass', 'eccentricity']])

```
array([[ 1.29      , 19.4       ,  0.231     ],
       [ 1.54      , 11.2       ,  0.08      ],
       [ 0.83      ,  4.8       ,  0.        ],
       ...,
       [ 5.83796389,  0.3334    ,  0.31      ],
       [ 5.83796389,  0.4       ,  0.27      ],
       [ 5.83796389,  0.42      ,  0.16      ]])
```

Машинне навчання з scikit-learn

```
from sklearn.impute import KNNImputer
KNNImputer().fit_transform(planet_data[['semimajoraxis',        'mass',
'eccentricity']])
```

```
array([[ 1.29    , 19.4     ,  0.231   ],
       [ 1.54    , 11.2     ,  0.08    ],
       [ 0.83    ,  4.8     ,  0.      ],
       ...,
       [ 0.404726,  0.3334  ,  0.31    ],
       [ 0.85486 ,  0.4     ,  0.27    ],
       [ 0.15324 ,  0.42    ,  0.16    ]])
```

Машинне навчання з scikit-learn

```
from sklearn.preprocessing import FunctionTransformer
FunctionTransformer(np.abs,
validate=True).fit_transform(X_train.dropna())
```

```
array([[1.45   , 0.51   , 4.94   ],
       [0.85   , 0.17   , 0.64   ],
       [1.192  , 0.08   , 0.03727],
       ...,
       [1.8    , 0.295  , 4.46   ],
       [0.0087 , 0.34   , 0.0652 ],
       [0.5    , 0.3    , 1.26   ]])
```

Машинне навчання з scikit-learn

Якщо наявні функції різних типів даних, можна використовувати клас ColumnTransformer для відображення перетворень у стовпець (або групу стовпців) за один виклик:

```
from sklearn.compose import ColumnTransformer
ColumnTransformer([('impute', KNNImputer(), [0]),('standard_scale', StandardScaler(), [1]),('min_max', MinMaxScaler(), [2])]).fit_transform(X_train)[:10]
```

```
array([[2.46002638e+00,             nan,             nan],
       [2.46002638e+00,             nan,             nan],
       [2.46002638e+00,             nan, 2.28055906e-05],
       [2.46002638e+00,             nan,             nan],
       [2.46002638e+00,             nan,             nan],
       [2.46002638e+00,             nan,             nan],
       [9.83346000e+00,             nan,             nan],
       [1.45000000e+00, 1.95196592e+00, 1.71374569e-03],
       [2.46002638e+00,             nan,             nan],
       [2.46002638e+00,             nan,             nan]])
```

Машинне навчання з scikit-learn

scikit-learn пропонує можливість створювати конвеєри, щоб спростити попередню обробку та гарантувати, що навчальні набори та тестувальні набори обробляються однаково.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
Pipeline([('scale', StandardScaler()), ('lr', LinearRegression())])
```

```
Pipeline(steps=[('scale', StandardScaler()), ('lr', LinearRegression())])
```

```
from sklearn.pipeline import make_pipeline
make_pipeline(StandardScaler(), LinearRegression())
```

```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('linearregression', LinearRegression())])
```

Машинне навчання з scikit-learn

```
ColumnTransformer([('impute', Pipeline([('impute',
KNNImputer()),('scale', StandardScaler())]), [0]),('standard_scale',
StandardScaler(), [1]),('min_max', MinMaxScaler(),
[2])]).fit_transform(X_train)[5:10]
```

```
array([[ 0.00000000e+00,             nan,             nan],
       [ 2.26820578e+00,             nan,             nan],
       [-3.10702964e-01,  1.95196592e+00,  1.71374569e-03],
       [ 0.00000000e+00,             nan,             nan],
       [ 0.00000000e+00,             nan,             nan]])
```

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]
```

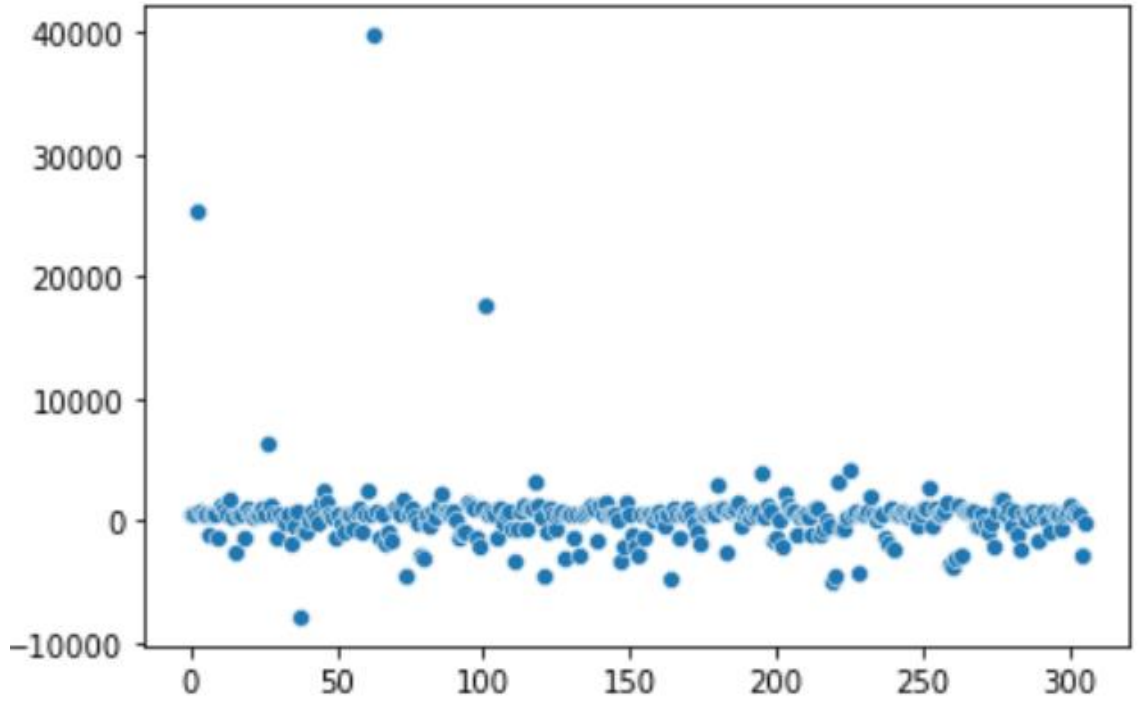
0.9692104355988059

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.

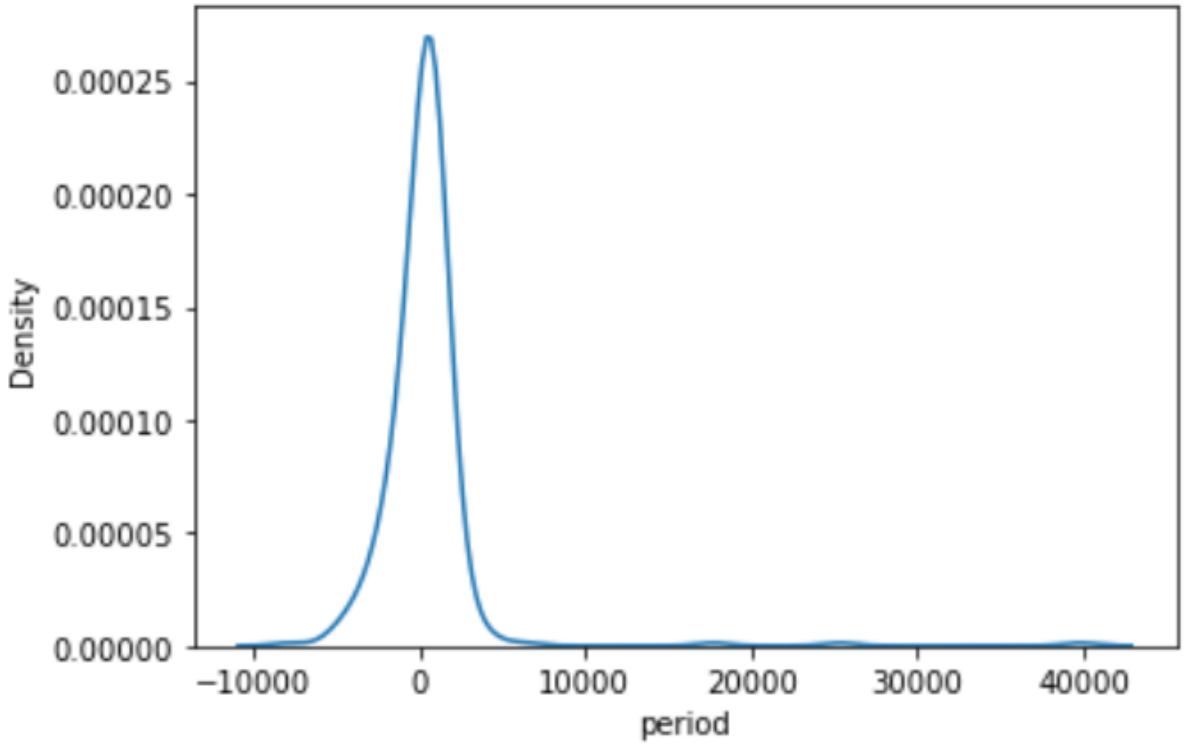
Машинне навчання з scikit-learn

```
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)
```

0.922014218429371

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

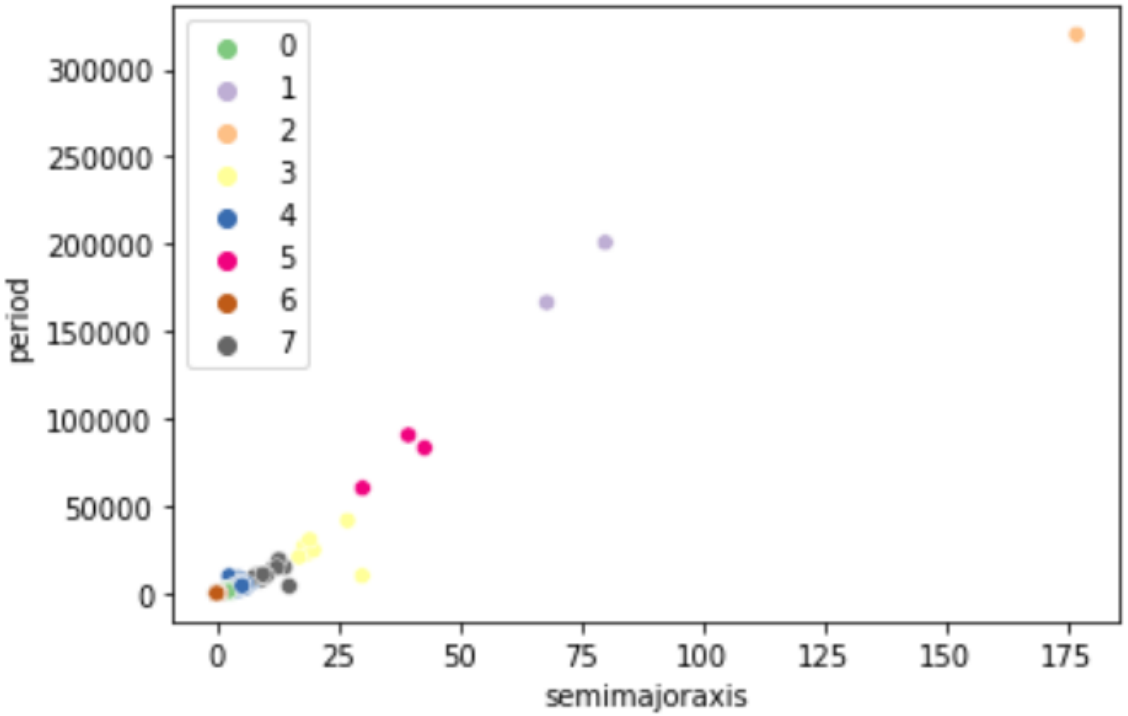
Оскільки алгоритм обрає початкові центри випадково, то можна отримати різний результат при різних запусках. Щоб уникнути цього, можна вставити значення параметру random_state.

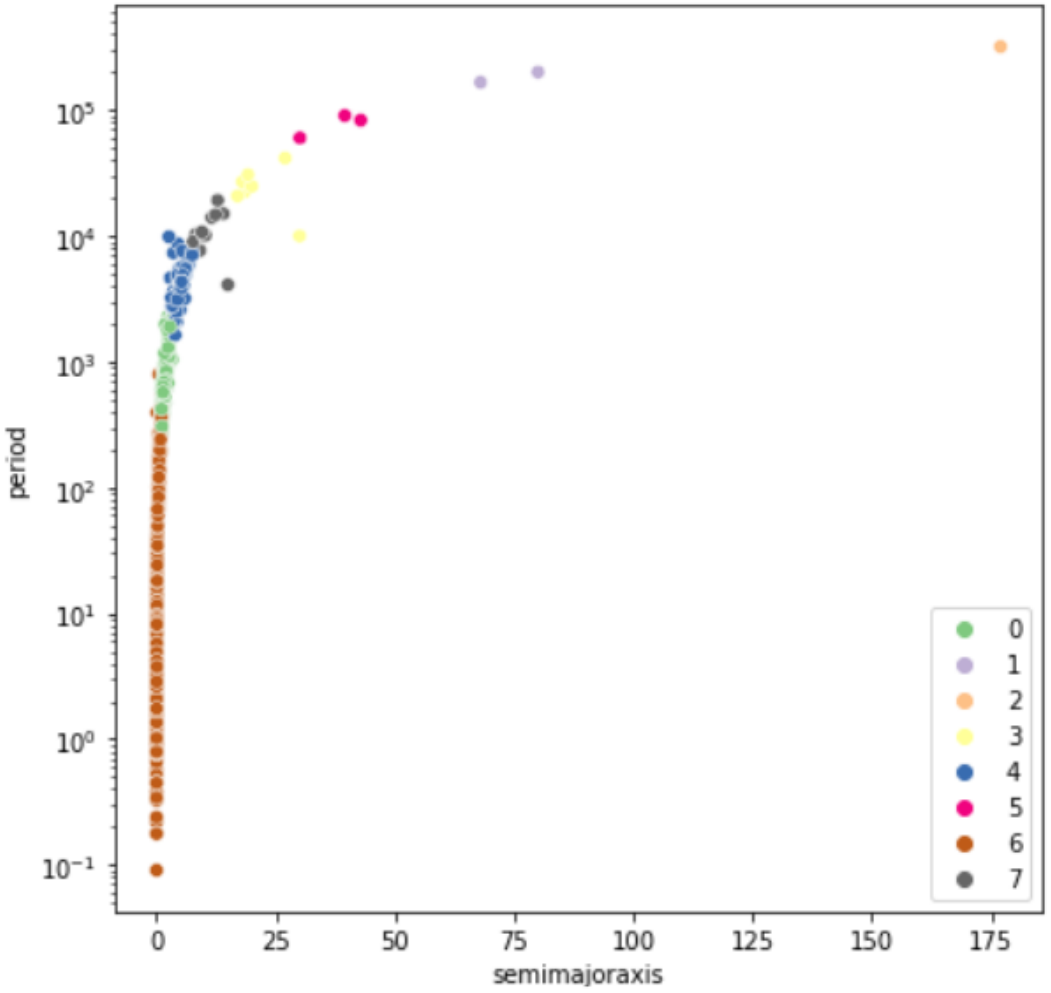
```
from sklearn.cluster import KMeans
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
kmeans_pipeline = Pipeline([('scale', StandardScaler()),('kmeans', KMeans(8, random_state=0))])
kmeans_data = planet_data[['semimajoraxis', 'period']].dropna()
kr
```

```
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