Машинне навчання з scikit-learn

Метою класифікації є визначення способу позначення даних за допомогою набору дискретних міток. Вона відрізняється від кластеризації при навчанні з учителем тим, що немає значення, наскільки близькі члени груп у просторі.

Одним з методі класифікації є логістична регресія, яка використовує логістичну сигмоїдну функцію для визначення ймовірностей у діапазоні [0, 1], які можна зіставити з мітками класів.

Машинне навчання з scikit-learn

Використаємо задачу класифікації для прогнозування класу якості червоного вина, виходячи з його хімічних властивостей.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

Машинне навчання з scikit-learn

df.info()

```
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
```

df.describe().T

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1599.0	8.319637	1.741096	4.60000	7.1000	7.90000	9.200000	15.90000
volatile acidity	1599.0	0.527821	0.179060	0.12000	0.3900	0.52000	0.640000	1.58000
citric acid	1599.0	0.270976	0.194801	0.00000	0.0900	0.26000	0.420000	1.00000
residual sugar	1599.0	2.538806	1.409928	0.90000	1.9000	2.20000	2.600000	15.50000
chlorides	1599.0	0.087467	0.047065	0.01200	0.0700	0.07900	0.090000	0.61100
free sulfur dioxide	1599.0	15.874922	10.460157	1.00000	7.0000	14.00000	21.000000	72.00000
total sulfur dioxide	1599.0	46.467792	32.895324	6.00000	22.0000	38.00000	62.000000	289.00000
density	1599.0	0.996747	0.001887	0.99007	0.9956	0.99675	0.997835	1.00369
pH	1599.0	3.311113	0.154386	2.74000	3.2100	3.31000	3.400000	4.01000
sulphates	1599.0	0.658149	0.169507	0.33000	0.5500	0.62000	0.730000	2.00000
alcohol	1599.0	10.422983	1.065668	8.40000	9.5000	10.20000	11.100000	14.90000
quality	1599.0	5.636023	0.807569	3.00000	5.0000	6.00000	6.000000	8.00000

Машинне навчання з scikit-learn

Оскільки вин високої якості дуже мало, створимо окремий стовпець-індикатор для цього класу:

```
df['high_quality'] = pd.cut(df.quality, bins=[0, 6, 10], labels=[0, 1])
df.high_quality.value_counts(normalize=True)
```

```
0    0.86429
1    0.13571
```

Машинне навчання з scikit-learn

Тепер потрібно розділити дані на навчальний та тестовий набір таким чином, щоб вони отримали однаковий відсоток вин високої якості (приблизно 14%):

```
from sklearn.model_selection import train_test_split
df_y = df.pop('high_quality')
df_X = df.drop(columns='quality')
X_train, X_test,y_train, y_test = train_test_split(df_X, df_y, test_size=0.1, random_state=0,stratify=df_y)
```

Машинне навчання з scikit-learn

Створюємо конвеєр.

Вагові коефіцієнти класів визначають, наскільки модель буде оштрафована за неправильні прогнози для кожного класу. При виборі збалансованих ваг, неправильні прогнози для менших класів матимуть більшу вагу, де вага буде обернено пропорційна частоті класу в даних.

```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
wine_quality_lr = Pipeline([('scale', StandardScaler()),('lr', LogisticRegression(class_weight='balanced', random_state=0))])
```

Машинне навчання з scikit-learn

Тренуємо модель:

```
wine_quality_lr.fit(X_train, y_train)
```

І використовуємо її для прогнозування:

```
quality_preds = wine_quality_lr.predict(X_test)
```

Машинне навчання з scikit-learn

Оцінити ефективність моделей класифікації можна виходячи з того, наскільки добре кожен клас у даних був передбачений моделлю. У випадку бінарної класифікації, тобто коли мітки лише дві, виділяють позитивний та негативний класи. Якщо міток більше, тоді один клас виділяють як позитивний, а інші – як негативні. Позитивний клас — це клас, який цікавить; всі інші класи вважаються негативними. У класифікації червоних вин позитивний клас – висока якість, а негативний – інші варіанти.

Машинне навчання з scikit-learn

Проблему класифікації можна оцінити, порівнявши передбачені мітки з фактичними мітками за допомогою матриці невідповідностей:

Істинна належність Результати класифікації	Позитивний	Негативний
Позитивний	TP	FP
Негативний	FN	TN

Де TP (True Positive) — кількість правильно визначених «позитивних» об’єктів;

FP (False Positive) — кількість «негативних» об’єктів, які хибно класифіковані як «позитивні»;

FN (False Negative) — кількість «позитивних» об’єктів, які хибно класифіковані як «негативні»;

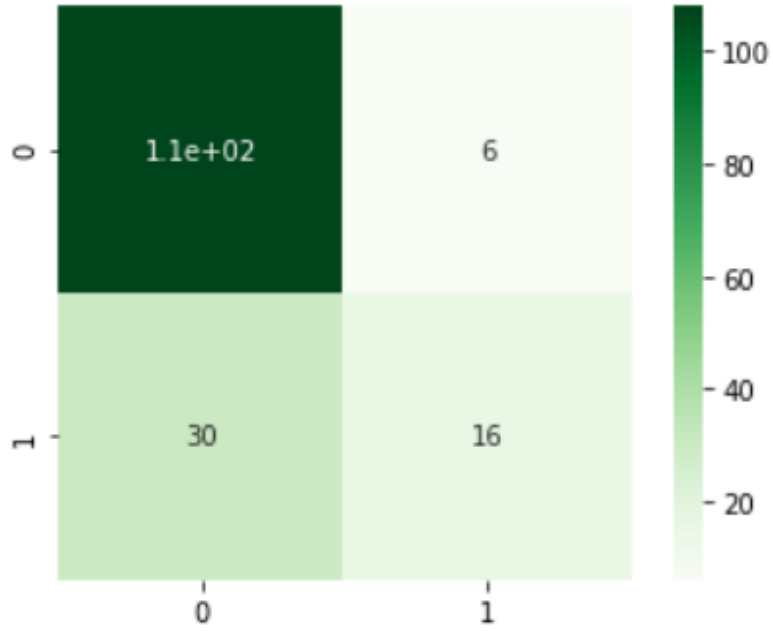
TN (True Negative) — кількість правильно визначених «негативних» об’єктів.

Машинне навчання з scikit-learn

Scikit-learn надає функцію confusion_matrix(), яку можна поєднати з функцією heatmap() від seaborn, щоб візуалізувати матрицю невідповідностей.

```
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(y_test, quality_preds)
```

```
array([[108,  30],
       [  6,  16]], dtype=int64)
```



Машинне навчання з scikit-learn

Використовуючи значення в матриці невідповідностей, можна обчислити показники, які допоможуть оцінити ефективність класифікатора. Найкращі показники залежатимуть від мети, для якої будується модель, і від того, чи збалансовані класи.

Коли класи приблизно однакові за розміром, можна використовувати точність, яка дасть відсоток правильно класифікованих значень:

$$A = \frac{TP + TN}{TP + FP + TN + FN}$$

```
wine_quality_lr.score(X_test, y_test)
```

0.775

Машинне навчання з scikit-learn

Частку неправильно класифікованих об’єктів можна отримати як:

```
from sklearn.metrics import zero_one_loss
zero_one_loss(y_test, quality_preds)
```

0.2249999999999998

Коли наявний дисбаланс класів, точність може стати ненадійним показником для вимірювання продуктивності. Тоді використовується влучність – це відношення правильно визначених позитивних результатів до всього, що позначено позитивним:

$$P = \frac{TP}{TP + FP}$$

Машинне навчання з scikit-learn

Також використовується повнота (чутливість), що дає відношення правильно визначених позитивних об’єктів до всіх позитивних об’єктів:

$$R = \frac{TP}{TP + FN}$$

Scikit-learn має функцію classification_report(), яка обчислює влучність і повноту. На додаток до обчислення цих показників для мітки класу, вона також обчислює незважене середнє між класами і середньозважене (середнє між класами, зважене за кількістю об’єктів у кожному класі). Стовпець support вказує кількість об’єктів, які належать до кожного класу, використовуючи позначені дані.

Машинне навчання з scikit-learn

Оцінка F1 допомагає збалансувати влучність і повноту, використовуючи середнє гармонійне з двох:

$$F_1 = \frac{2 \times TP}{2 \times TP + FP + FN}$$

```
from sklearn.metrics import classification_report
print(classification_report(y_test, quality_preds))
```

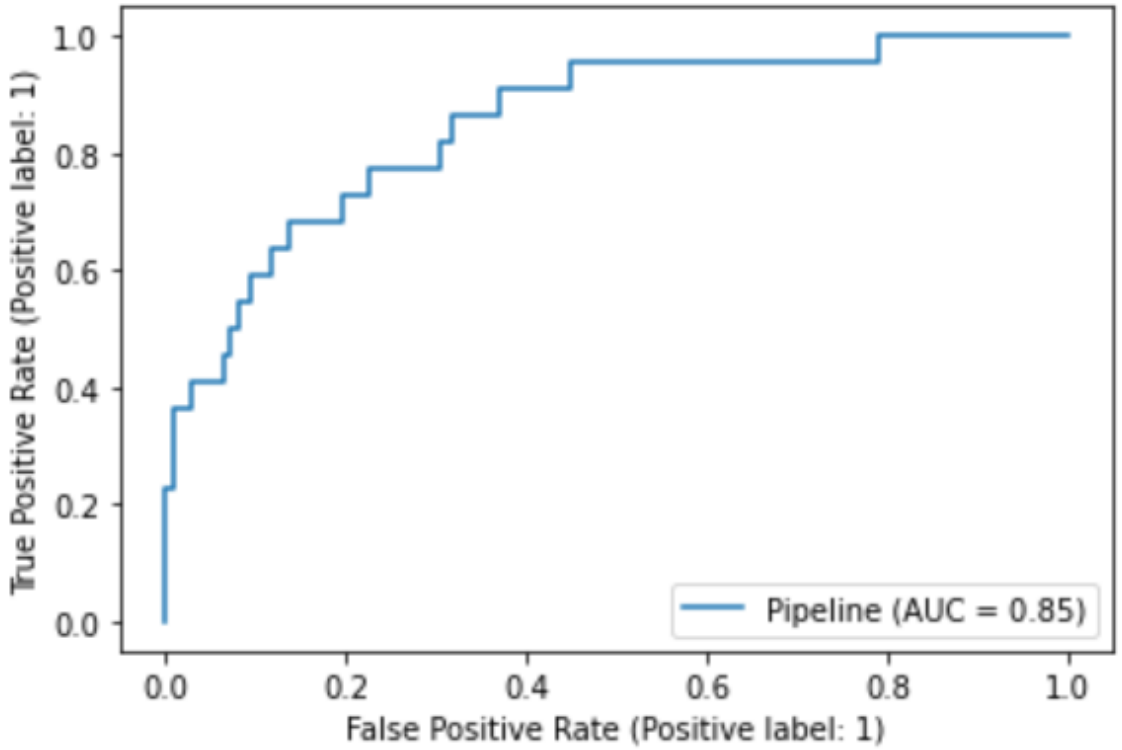
```
              precision    recall  f1-score   support

           0       0.95      0.78      0.86       138
           1       0.35      0.73      0.47        22

    accuracy                           0.78       160
   macro avg       0.65      0.75      0.66       160
weighted avg       0.86      0.78      0.80       160
```

Машинне навчання з scikit-learn

Крива AUC-ROC є інструментом для вимірювання та оцінки продуктивності моделей класифікації. ROC (Receiver Operating Characteristics) — це графічна візуалізація характеристик моделі. Вона будує двовимірний ймовірнісний графік між частотою хибно позитивних (або 1-частота правильно негативних) і частотою правильних позитивних (або чутливістю). Також можна представити площу, охоплену моделлю, одним числом за допомогою AUC (площа під кривою).

Машинне навчання з scikit-learn

```
from sklearn.metrics import plot_roc_curve
plot_roc_curve(wine_quality_lr,X_test, y_test)
```



Машинне навчання з scikit-learn

Після створення та оцінки моделі, варто її налаштувати та обрати найкращу. Параметри моделі називаються гіперпараметрами.

Scikit-learn має клас GridSearchCV в модулі model_selection для налаштування моделі. GridSearch дозволяє визначити простір пошуку та перевірити всі комбінації гіперпараметрів у цьому просторі, зберігаючи ті, які призводять до найкращої моделі.Подібні класи використовують перехресну перевірку, тобто вони ділять навчальні дані на підмножини, деякі з яких будуть набором для оцінки моделі (без потреби в тестових дани допоки модель не буде навчена).

Машинне навчання з scikit-learn

Одним із поширених методів перехресної перевірки є k-кратна перехресна перевірка, яка розбиває навчальні дані на k підмножин і тренує модель k разів, щоразу залишаючи одну підмножину для тестування. Оцінка моделі буде середнім значенням для k тестових наборів.

Машинне навчання з scikit-learn

Щоб використовувати GridSearchCV, потрібно надати модель (або конвеєр) і простір пошуку, який буде словником, що відображає гіперпараметр для налаштування (за назвою) на список значень, які потрібно спробувати. За бажанням, можна надати для використання метрику оцінки, а також значення k для використання при перехресній перевірці.

Машинне навчання з scikit-learn

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
pipeline = Pipeline([('scale', MinMaxScaler()),('lr', LogisticRegression(class_weight='balanced',random_state=0))])
search_space = {'lr__C': np.logspace(-1, 1, num=10),'lr__fit_intercept': [True, False]}
lr_grid = GridSearchCV(pipeline, search_space, scoring='f1_macro', cv=5).fit(X_train, y_train)
```

Можна переглянути можливі значення scoring за допомогою

```
from sklearn.metrics import SCORERS
SCORERS.keys()
```

Машинне навчання з scikit-learn

Можна подивидитись найкращі значення параметрів за допомогою атрибуту best_params_

```
lr_grid.best_params_
```

{'lr__C': 3.593813663804626, 'lr__fit_intercept': True}

```
lr_grid.best_score_
```

0.6854701118156911

```
print(classification_report(y_test, lr_grid.predict(X_test)))
```

	precision	recall	f1-score	support
0	0.94	0.80	0.87	138
1	0.36	0.68	0.47	22
accuracy			0.79	160
macro avg	0.65	0.74	0.67	160
weighted avg	0.86	0.79	0.81	160

Машинне навчання з scikit-learn

```
from sklearn.model_selection import RepeatedStratifiedKFold
lr_grid = GridSearchCV(pipeline, search_space,
scoring='f1_macro',cv=RepeatedStratifiedKFold(random_state=0)).fit(X
_train, y_train)
lr_grid.best_score_
```

0.689078602044416

```
lr_grid.best_params_
```

{'lr__C': 5.994842503189409, 'lr__fit_intercept': True}

Машинне навчання з scikit-learn

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import make_scorer, mean_squared_error
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
model_pipeline = Pipeline([('scale', StandardScaler()),('lr',
LinearRegression())])
search_space = {'scale__with_mean': [True, False],'scale__with_std':
[True, False],'lr__fit_intercept': [True, False],'lr__normalize': [True,
False]}
grid = GridSearchCV(model_pipeline, search_space,
cv=5,scoring={'r_squared': 'r2','mse': 'neg_mean_squared_error','mae':
'neg_mean_absolute_error','rmse': make_scorer(lambda x, y: -
np.sqrt(mean_squared_error(x, y)))}, refit='mae').fit(X_train, y_train)
```

Машинне навчання з scikit-learn

grid.best_score_

-1215.9916625290548

grid.best_params_

{'lr__fit_intercept': False,
 'lr__normalize': True,
 'scale__with_mean': False,
 'scale__with_std': False}

Машинне навчання з scikit-learn

З метою підвищити продуктивність моделі можна також розглянути способи надання найкращих ознак (вхідних даних моделі) для моделі через процес розробки ознак.

Створення нових ознак зі старих

Вибір ознак — це процес визначення, за якими ознаками слід тренувати модель. Це можна зробити вручну або за допомогою іншого процесу, наприклад машинного навчання. Моделі, створені за багатьма ознаками, складні, але, на жаль, мають більшу тенденцію до перенавчання. Це називається прокляттям розмірності.

Машинне навчання з scikit-learn

Вилучення ознак — це ще один спосіб подолати прокляття розмірності. Під час вилучення ознак зменшується розмірність даних, створюючи комбінації ознак за допомогою перетворення. Ці нові ознаки можна використовувати замість початкових, тим самим зменшуючи розмірність проблеми. Цей процес, який називається зменшенням розмірності, також включає прийоми, за допомогою яких відшукується певна кількість компонентів, які пояснюють більшу частину дисперсії в даних.

Машинне навчання з scikit-learn

Деякі ознаки впливають на результат не окремо, а в поєднанні. Щоб врахувати такий ефект, потрібно додати нову ознаку взаємодії, яка буде, наприклад, добутком цих ознак.

Також можна збільшити вплив ознаки в моделі, створивши нову ознаку як піднесену до степеня стару.

Все це можна зробити за допомогою класу PolynomialFeatures в модулі preprocessing.

```
from sklearn.preprocessing import PolynomialFeatures
PolynomialFeatures(degree=2).fit_transform(X_train[['citric acid', 'fixed acidity']])
```

Машинне навчання з scikit-learn

Значення параметру degree=2, застосоване до ознак x та y, створить поліном:

$1+x+y+x^2+xy+y^2$

```
array([[1.000e+00, 5.500e-01, 9.900e+00, 3.025e-01, 5.445e+00, 9.801e+01],
       [1.000e+00, 4.600e-01, 7.400e+00, 2.116e-01, 3.404e+00, 5.476e+01],
       [1.000e+00, 4.100e-01, 8.900e+00, 1.681e-01, 3.649e+00, 7.921e+01],
       ...,
       [1.000e+00, 1.200e-01, 7.000e+00, 1.440e-02, 8.400e-01, 4.900e+01],
       [1.000e+00, 3.100e-01, 7.600e+00, 9.610e-02, 2.356e+00, 5.776e+01],
       [1.000e+00, 2.600e-01, 7.700e+00, 6.760e-02, 2.002e+00, 5.929e+01]])
```

Створимо модель з такими ознаками

pipeline = Pipeline([('poly', PolynomialFeatures(degree=2)),('scale', MinMaxScaler()),('lr', LogisticRegression(class_weight='balanced', random_state=0))]).fit(X_train, y_train)

Машинне навчання з scikit-learn

Порівняємо її характеристики з попередніми:

```
preds = pipeline.predict(X_test)
print(classification_report(y_test, preds))
```

	precision	recall	f1-score	support
0	0.95	0.79	0.86	138
1	0.36	0.73	0.48	22
accuracy			0.78	160
macro avg	0.65	0.76	0.67	160
weighted avg	0.87	0.78	0.81	160

Попередня модель:

	precision	recall	f1-score	support
0	0.95	0.78	0.86	138
1	0.35	0.73	0.47	22
accuracy			0.78	160
macro avg	0.65	0.75	0.66	160
weighted avg	0.86	0.78	0.80	160

Машинне навчання з scikit-learn

Інколи краще не додавати нових ознак до набору даних, а навпаки, зменшити їхню кількість. Можна просто обрати підмножину ознак, але можна спробувати зберегти інформацію, що міститься в ознаках.

Однією з поширених стратегій вибору ознак є відкидання ознак з низькою дисперсією. Ці ознаки не надто інформативні, оскільки вони в основному мають однакове значення в усьому наборі даних. Scikit-learn має клас VarianceThreshold для здійснення вибору ознак відповідно до мінімального порога дисперсії. За замовчуванням він відкидає будь-які ознаки, які мають нульову дисперсію; однак можна задати власний поріг.

Машинне навчання з scikit-learn

```
from sklearn.feature_selection import VarianceThreshold
pipeline  = Pipeline([('feature_selection',VarianceThreshold()),('scale', StandardScaler()),('lr', LogisticRegression(class_weight='balanced’,random_state=0))]).fit(X_train, y_train)

X_train.columns[~pipeline.named_steps['feature_selection'].get_support()]
Index([], dtype='object')
pipeline  = Pipeline([('feature_selection',VarianceThreshold(threshold=0.01)),('scale', StandardScaler()),('lr', LogisticRegression(class_weight='balanced’,random_state=0))]).fit(X_train, y_train)
Index(['chlorides', 'density'], dtype='object')
```

Машинне навчання з scikit-learn

classification_report(y_test, preds)

	precision	recall	f1-score	support
0	0.94	0.82	0.88	138
1	0.38	0.68	0.48	22
accuracy			0.80	160
macro avg	0.66	0.75	0.68	160
weighted avg	0.86	0.80	0.82	160

Попередні моделі:

	precision	recall	f1-score	support
0	0.95	0.78	0.86	138
1	0.35	0.73	0.47	22
accuracy			0.78	160
macro avg	0.65	0.75	0.66	160
weighted avg	0.86	0.78	0.80	160

	precision	recall	f1-score	support
0	0.95	0.79	0.86	138
1	0.36	0.73	0.48	22
accuracy			0.78	160
macro avg	0.65	0.76	0.67	160
weighted avg	0.87	0.78	0.81	160

Машинне навчання з scikit-learn

Якщо є підстави вважати, що всі ознаки мають цінність, можна використати видобуток ознак, а не відкидання. Аналіз головних компонентів (PCA) виконує виділення ознак, проектуючи дані високої розмірності на нижчу розмірність, тим самим зменшуючи розмірність. Натомість отримуються n компонентів, які максимізують пояснену дисперсію.

```
from sklearn.decomposition import PCA
pipeline = Pipeline([('normalize', MinMaxScaler()),('pca', PCA(4,
random_state=0)),('lr', LogisticRegression(class_weight='balanced',
random_state=0))]).fit(X_train, y_train)
```

Машинне навчання з scikit-learn

classification_report(y_test, preds)

	precision	recall	f1-score	support
0	0.93	0.83	0.88	138
1	0.38	0.64	0.47	22
accuracy			0.81	160
macro avg	0.66	0.73	0.68	160
weighted avg	0.86	0.81	0.83	160

Попередня модель:

	precision	recall	f1-score	support
0	0.95	0.78	0.86	138
1	0.35	0.73	0.47	22
accuracy			0.78	160
macro avg	0.65	0.75	0.66	160
weighted avg	0.86	0.78	0.80	160

Машинне навчання з scikit-learn

Може бути потреба побудувати модель на основі ознак з різних джерел, таких як PCA, на додаток до вибору підмножини ознак. Для цих цілей scikit-learn має клас FeatureUnion в модулі pipeline.

```
from sklearn.pipeline import FeatureUnion, Pipeline
combined_features = FeatureUnion([('variance',
VarianceThreshold(threshold=0.01)),('poly',
PolynomialFeatures(degree=2, include_bias=False,
interaction_only=True))])
pipeline = Pipeline([('normalize', MinMaxScaler()),('feature_union',
combined_features),('lr', LogisticRegression(class_weight='balanced',
random_state=0))]).fit(X_train, y_train)
```

Машинне навчання з scikit-learn

classification_report(y_test, preds)

	precision	recall	f1-score	support
0	0.94	0.80	0.87	138
1	0.36	0.68	0.47	22
accuracy			0.79	160
macro avg	0.65	0.74	0.67	160
weighted avg	0.86	0.79	0.81	160

Попередня модель:

	precision	recall	f1-score	support
0	0.95	0.78	0.86	138
1	0.35	0.73	0.47	22
accuracy			0.78	160
macro avg	0.65	0.75	0.66	160
weighted avg	0.86	0.78	0.80	160

Машинне навчання з scikit-learn

Існують моделі машинного навчання, що допомагають визначити, які ж ознаки є важливими.

Дерева рішень рекурсивно розбивають дані, приймаючи рішення про те, які ознаки використовувати для кожного поділу. Вони використовують «жадібний» підхід, тобто вони щоразу шукають найбільший розкол, який можуть зробити; він не обов’язково є оптимальним. Можна використовувати дерево рішень, щоб оцінити важливість ознак, які визначають, як дерево розбиває дані на вузлах прийняття рішень.

Дерева рішень не потребують масштабування даних.

Машинне навчання з scikit-learn

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state=0).fit(X_train, y_train)
X_train.columns
```

'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar’,
'chlorides', 'free sulfur dioxide', 'total sulfur dioxide’,
'density', 'pH', 'sulphates', 'alcohol'

```
dt.feature_importances_
```

0.07456392, 0.1116417 , 0.05636227, 0.08647322,
0.04320703, 0.06846924, 0.09145632,
0.03864031,0.09035411, 0.10572727, 0.2331046

Машинне навчання з scikit-learn

Можна також перевірити важливість ознак в даних про планети:

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor(random_state=0).fit(X_train, y_train)
```

```
[('semimajoraxis', 0.9969449557611615),
 ('mass', 0.001538098626057415 4),
 ('eccentricity', 0.0015169456127809738)]
```

Машинне навчання з scikit-learn

Дерева рішень можна використовувати і для класифікації:

```
dt = DecisionTreeClassifier(class_weight='balanced',random_state=0).fit(X_train, y_train)
```

	precision	recall	f1-score	support
0	0.94	0.94	0.94	138
1	0.62	0.59	0.60	22
accuracy			0.89	160
macro avg	0.78	0.77	0.77	160
weighted avg	0.89	0.89	0.89	160

	precision	recall	f1-score	support
0	0.95	0.78	0.86	138
1	0.35	0.73	0.47	22
accuracy			0.78	160
macro avg	0.65	0.75	0.66	160
weighted avg	0.86	0.78	0.80	160

Машинне навчання з scikit-learn

Гіперпараметри моделі дерев рішень:
criterion{“gini”, “entropy”} – функція, за якою вимірюється розгалуження
splitter{“best”, “random”} – яким чином обирається розгалуження в кожному вузлі
max_depth – максимальна глибина дерева
min_samples_split - мінімальна кількість вибірок, необхідних для розгалуження у внутрішньому вузлі
min_samples_leaf - мінімальна кількість вибірок, яка повинна бути у листку
max_features – кількість ознак, що розглядаються для пошуку кращого розгалуження
max_leaf_nodes – максимальна кількість листків

Машинне навчання з scikit-learn

Дерева рішень схильні до перенавчання, коли модель є занадто складною і повністю підлаштовується під навчальні дані, але погано підходить для узагальнення і працює з новими даними.

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor(random_state=0).fit(X_train, y_train)

dt.score(X_train, y_train)
1.0

dt.score(X_test, y_test)
0.882439213411108
```

Машинне навчання з scikit-learn

Щоб уникнути перенавчання можна обмежити розмір дерева.

```
dt = DecisionTreeRegressor(max_depth=4,random_state=0).fit(X_train, y_train)
```

```
dt.score(X_train, y_train)
```
0.985850396292969 8

```
dt.score(X_test, y_test)
```
0.923251287160794 5

Також дерева не здатні виконувати екстраполяцію.

Машинне навчання з scikit-learn

Ансамбль методів поєднує багато моделей (часто слабких), щоб створити сильнішу, яка або мінімізує середню похибку між спостережуваними і прогнозованими значеннями (зміщення), або покращить, наскільки добре вона узагальнює незнайомі дані (мінімізує дисперсію).

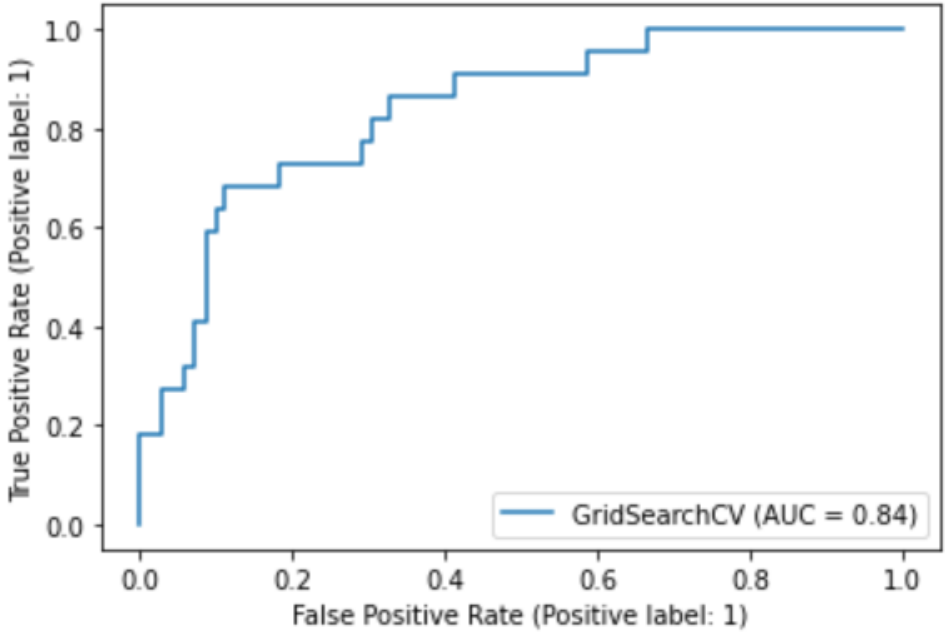
Дерева рішень мають тенденцію до перенавчання, особливо якщо не встановлено обмежень на глибину зростання (за допомогою параметрів max_depth і min_samples_leaf). Цю проблему можна вирішити за допомогою випадкового лісу, де навчається багато дерев рішень паралельно і агрегується результат. Ліс складається з невеликих за глибиною дерев, що відрізняються одне від одного за рахунок випадкового обрання ознак для поділу та використання різних підмножин вхідних даних для різних дерев.

Машинне навчання з scikit-learn

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
rf = RandomForestClassifier(n_estimators=100, random_state=0)
search_space = {'max_depth': [4, 8],'min_samples_leaf': [4, 6]}
rf_grid = GridSearchCV(rf, search_space, cv=5,
scoring='precision').fit(X_train, y_train)
rf_grid.score(X_test, y_test)
```

0.6

	precision	recall	f1-score	support
0	0.89	0.97	0.93	138
1	0.60	0.27	0.37	22
accuracy			0.88	160
macro avg	0.75	0.62	0.65	160
weighted avg	0.85	0.88	0.85	160

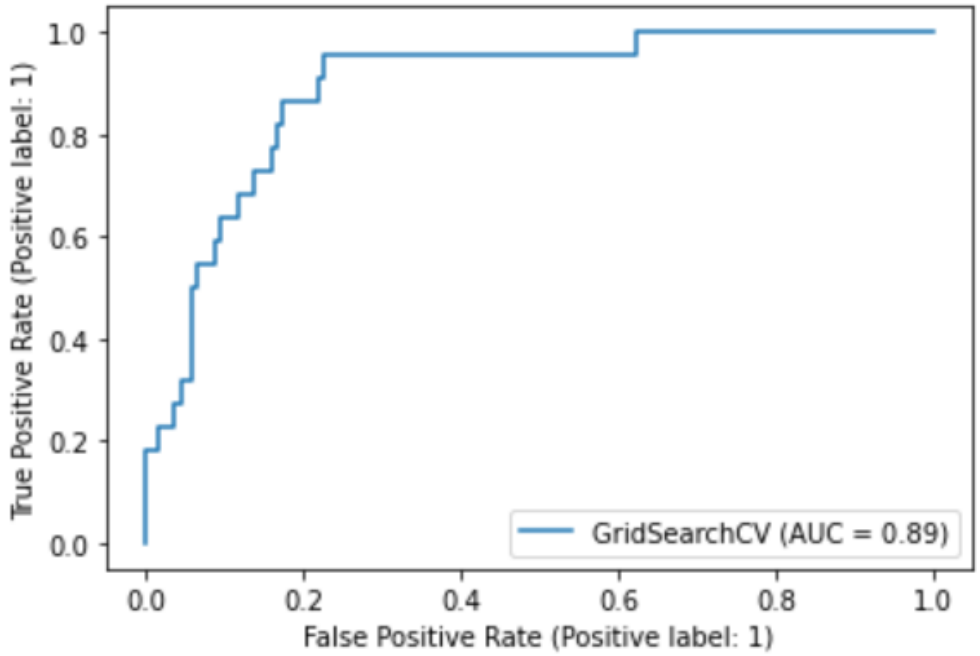


Машинне навчання з scikit-learn

```
rf = RandomForestClassifier(n_estimators=100, random_state=0)
search_space = {'max_depth': [4, 8],'min_samples_leaf': [4, 6]}
rf_grid = GridSearchCV(rf, search_space, cv=5,
scoring='roc_auc').fit(X_train, y_train)
rf_grid.score(X_test, y_test)
```

0.8913043478260869

	precision	recall	f1-score	support
0	0.90	0.96	0.93	138
1	0.54	0.32	0.40	22
accuracy			0.87	160
macro avg	0.72	0.64	0.66	160
weighted avg	0.85	0.87	0.85	160



Машинне навчання з scikit-learn

Бустинг намагається виправити помилки попередніх моделей. Один із способів зробити це – рухатися в напрямку найбільш крутого зниження функції втрат для моделі. Оскільки градієнт (узагальнення похідної з багатьма змінними) є напрямком найбільш крутого підйому, це можна зробити шляхом розрахунку негативного градієнта, який дає напрямок найбільш крутого спуску, що означає найкраще покращення функції втрат від поточного результату. Ця техніка називається градієнтним спуском.

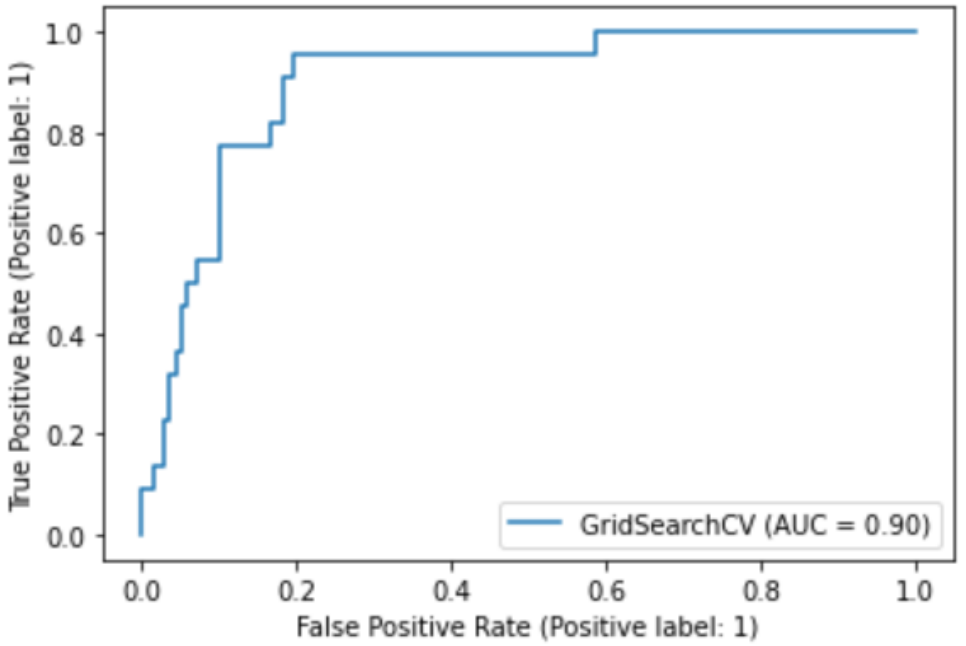
GradientBoostingClassifier також використовує багато дерев рішень, кожне з них будується таким чином, щоб зменшити помилку попередніх.

Машинне навчання з scikit-learn

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
gb = GradientBoostingClassifier(n_estimators=100, random_state=0)
search_space = {'max_depth': [4, 8], 'min_samples_leaf': [4,
6],'learning_rate': [0.1, 0.5, 1]}
gb_grid = GridSearchCV(gb, search_space, cv=5,
scoring='f1_macro').fit(X_train, y_train)
gb_grid.score(X_test, y_test)
```

0.7226024272287617

	precision	recall	f1-score	support
0	0.92	0.95	0.93	138
1	0.59	0.45	0.51	22
accuracy			0.88	160
macro avg	0.75	0.70	0.72	160
weighted avg	0.87	0.88	0.87	160



Машинне навчання з scikit-learn

Іноді важко знайти єдину модель, яка б добре працювала для всіх даних, тому є потреба оцінити різні моделі, щоб прийняти остаточне рішення. Scikit-learn має клас VotingClassifier для агрегування оцінок моделей відносно завдань класифікації. Можна вказати тип голосування, при значенні hard результат обрається за правилом більшості, а з soft передбачатиметься клас з найвищою сумою ймовірностей у моделях.

```
from sklearn.ensemble import VotingClassifier
majority_rules = VotingClassifier([('lr', lr_grid.best_estimator_),('rf',
rf_grid.best_estimator_),('gb',
gb_grid.best_estimator_)],voting='hard').fit(X_train, y_train)
max_probabilities = VotingClassifier([('lr', lr_grid.best_estimator_),('rf',
rf_grid.best_estimator_),('gb',
gb_grid.best_estimator_)],voting='soft').fit(X_train,y_train)
```

Машинне навчання з scikit-learn

print(classification_report(y_test, majority_rules.predict(X_test)))

	precision	recall	f1-score	support
0	0.92	0.95	0.93	138
1	0.59	0.45	0.51	22
accuracy			0.88	160
macro avg	0.75	0.70	0.72	160
weighted avg	0.87	0.88	0.87	160

	precision	recall	f1-score	support
0	0.95	0.78	0.86	138
1	0.35	0.73	0.47	22
accuracy			0.78	160
macro avg	0.65	0.75	0.66	160
weighted avg	0.86	0.78	0.80	160

Машинне навчання з scikit-learn

print(classification_report(y_test, max_probabilities.predict(X_test)))

	precision	recall	f1-score	support
0	0.92	0.93	0.92	138
1	0.52	0.50	0.51	22
accuracy			0.87	160
macro avg	0.72	0.71	0.72	160
weighted avg	0.87	0.87	0.87	160

	precision	recall	f1-score	support
0	0.95	0.78	0.86	138
1	0.35	0.73	0.47	22
accuracy			0.78	160
macro avg	0.65	0.75	0.66	160
weighted avg	0.86	0.78	0.80	160

Машинне навчання з scikit-learn

Для задачі класифікації також можна використовувати метод k-найближчих сусідів (k-NN), яка буде класифікувати спостереження відповідно до класу k-найближчих спостережень у n-вимірному просторі даних.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5).fit(X_train, y_train)
knn_preds = knn.predict(X_test)
```

	precision	recall	f1-score	support
0	0.91	0.93	0.92	138
1	0.50	0.41	0.45	22
accuracy			0.86	160
macro avg	0.70	0.67	0.69	160
weighted avg	0.85	0.86	0.86	160

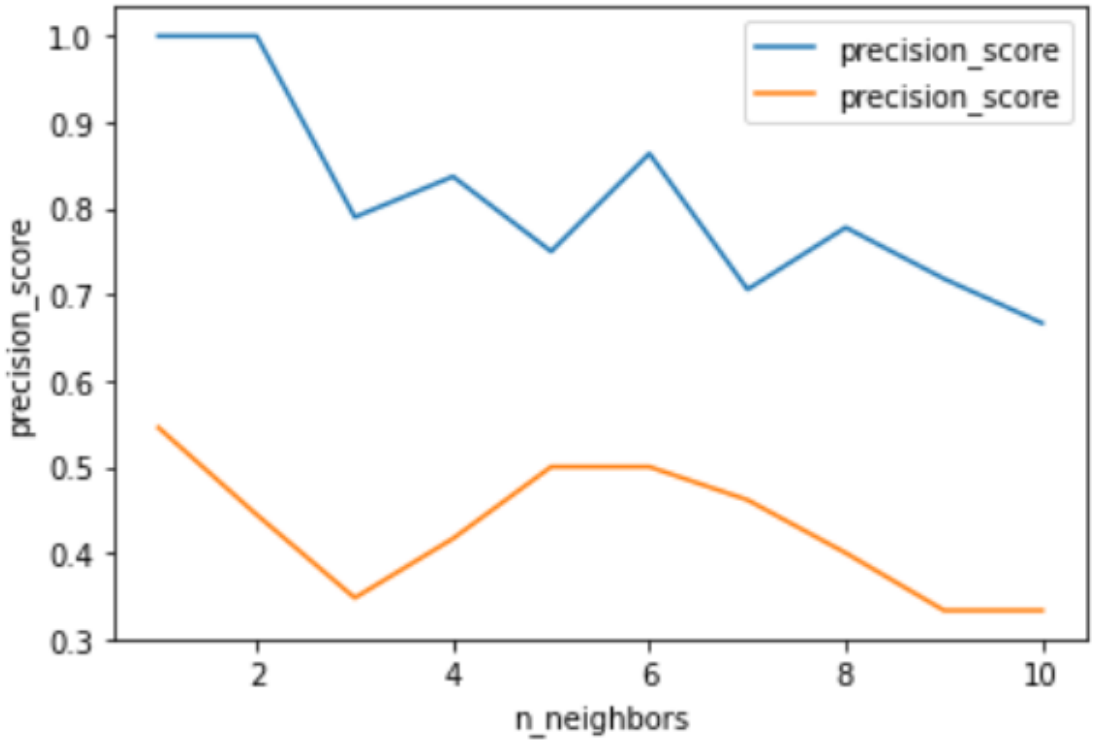
Машинне навчання з scikit-learn

Можна підібрати кількість найближчих сусідів, яка максимізує значення заданої метрики.

```
from sklearn.metrics import precision_score
training_p = []
test_p = []
neighbors_settings = range(1, 11)
for n_neighbors in neighbors_settings:
    knn = KNeighborsClassifier(n_neighbors=n_neighbors)
    knn.fit(X_train, y_train)
    training_p.append(precision_score(y_train,knn.predict(X_train)))
    test_p.append(precision_score(y_test,knn.predict(X_test)))
```

Машинне навчання з scikit-learn

Побудуємо графік:



Машинне навчання з scikit-learn

```
knn = KNeighborsClassifier(n_neighbors=1).fit(X_train, y_train)
precision_score(y_test,knn.predict(X_test))
```

0.545454545454545

```
knn = KNeighborsClassifier(n_neighbors=3).fit(X_train, y_train)
precision_score(y_test,knn.predict(X_test))
```

0.3478260869565217

Машинне навчання з scikit-learn

Використаємо ще один алгоритм, машини опорних векторів (SVM), який проектує дані у вищий вимір, щоб знайти гіперплощину, яка розділяє класи. Гіперплощина є n-вимірним еквівалентом площини, так само, як площина є двовимірним еквівалентом прямої. SVM, як правило, стійкі до викидів і можуть моделювати нелінійні межі рішень; однак SVM зазвичай працює повільніше.

```
from sklearn.svm import SVC
svc = SVC(gamma='auto', class_weight='balanced').fit(X_train, y_train)
svm_preds = svc.predict(X_test)
```

Машинне навчання з scikit-learn

print(classification_report(y_test, knn_preds))

	precision	recall	f1-score	support
0	0.93	0.93	0.93	138
1	0.55	0.55	0.55	22
accuracy			0.88	160
macro avg	0.74	0.74	0.74	160
weighted avg	0.88	0.88	0.88	160

print(classification_report(y_test, svm_preds))

	precision	recall	f1-score	support
0	0.94	0.86	0.90	138
1	0.42	0.64	0.51	22
accuracy			0.83	160
macro avg	0.68	0.75	0.70	160
weighted avg	0.87	0.83	0.84	160

Машинне навчання з scikit-learn

Регуляризація:

При використанні регресій, можна додавати так званий штрафний член до рівняння регресії, щоб зменшити перенавчання, «караючи» певні рішення за коефіцієнти, прийняті моделлю; це називається регуляризацією. Модель буде шукати коефіцієнти, які мінімізують цей штрафний член. Ідея полягає в тому, щоб зменшити коефіцієнти до нуля для функцій, які не сприяють зменшенню похибки моделі.

Машинне навчання з scikit-learn

L2-регуляризація (Ridge) «карає» високі коефіцієнти, додаючи суму квадратів коефіцієнтів до функції вартості (яку регресію намагається мінімізувати при підборі) відповідно до наступного штрафного члену.

$$L2 = \lambda \sum_j \hat{\beta}_j^{\,2}$$

λ вказує, наскільки великим буде штраф. Коли вона дорівнює нулю, маємо звичайну регресію найменших квадратів, як і раніше.

Машинне навчання з scikit-learn

Регуляризація L1 (Lasso) зводить коефіцієнти до нуля, додаючи суму абсолютних значень коефіцієнтів до функції вартості. Вона є більш надійною, ніж регуляризація L2, оскільки менш чутлива до екстремальних значень:

$$L1 = \lambda \sum_{j} |\hat{\beta}_j|$$

Оскільки L1 зводить коефіцієнти певних ознак у регресії до нуля (тобто вони не зроблять внесок у модель), кажуть, що вона виконує вибір ознак.

Машинне навчання з scikit-learn

Регуляризація еластична мережа (elastic net) поєднує обидва штрафні члени з L1 і L2 в наступний штрафний член, в якому можна налаштувати як силу покарання (λ), так і відсоток штрафу, який дорівнює L1 (і, отже, відсоток, який дорівнює L2) за допомогою α:

$$elastic\ net\ penalty = \lambda\left(\frac{1-\alpha}{2}\sum_j \hat{\beta}_j{}^2 + \alpha\sum_j |\hat{\beta}_j|\right)$$

Машинне навчання з scikit-learn

```
from sklearn.linear_model import Ridge, Lasso, ElasticNet
ridge, lasso, elastic = Ridge(), Lasso(), ElasticNet()
for model in [ridge, lasso, elastic]:
    model.fit(X_train, y_train)
    print(f'{model.__class__.__name__}: ' f'{model.score(X_test,
y_test):.4}’)
```

Ridge: 0.9206

Lasso: 0.9208

ElasticNet: 0.9047

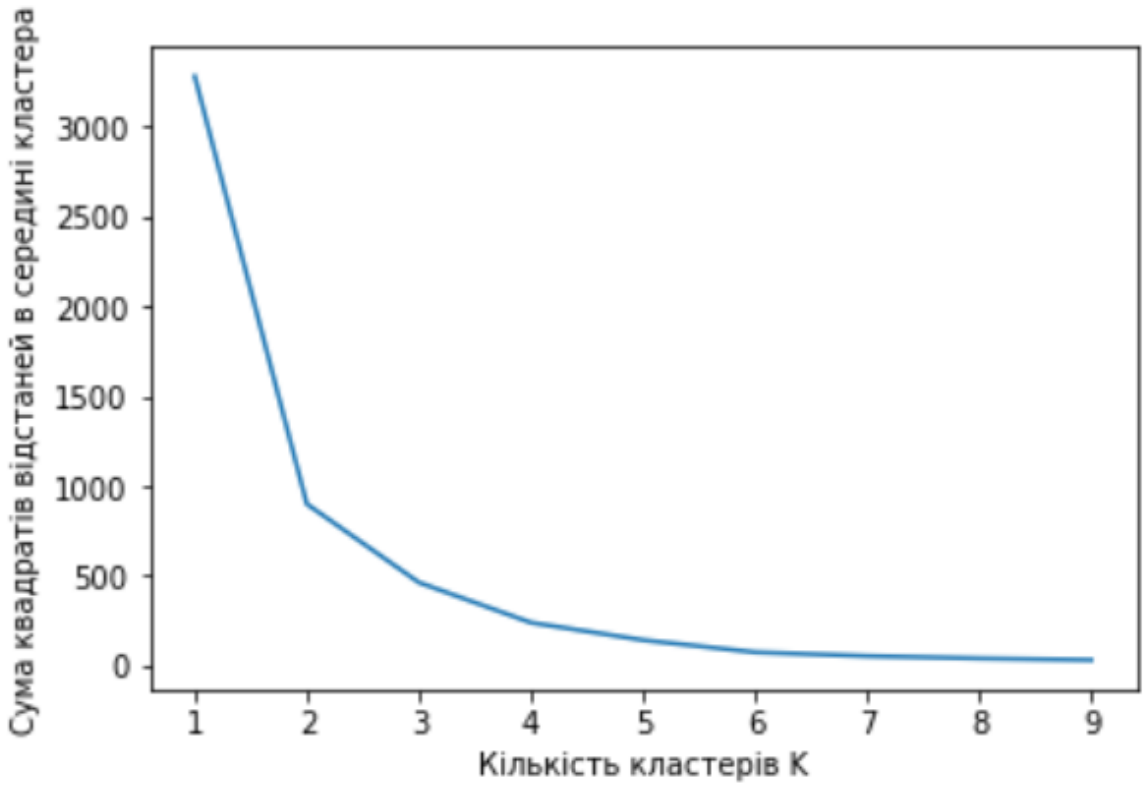
Машинне навчання з scikit-learn

Для тих алгоритмів кластеризації, які потребують попереднього задання кількості кластерів, можна підібрати оптимальну кількість.

```
d_list = []
for i in range(1, 10):
    kmeans_pipeline = Pipeline([('scale', StandardScaler()),('kmeans', KMeans(i, random_state=0))])
    kmeans_pipeline.fit(kmeans_data)
    d_list.append(kmeans_pipeline.named_steps['kmeans'].inertia_)
plt.plot(range(1, 10), d_list)
plt.xlabel('Кількість кластерів K')
plt.ylabel('Сума квадратів відстаней в середині кластера')
plt.show()
```

Машинне навчання з scikit-learn

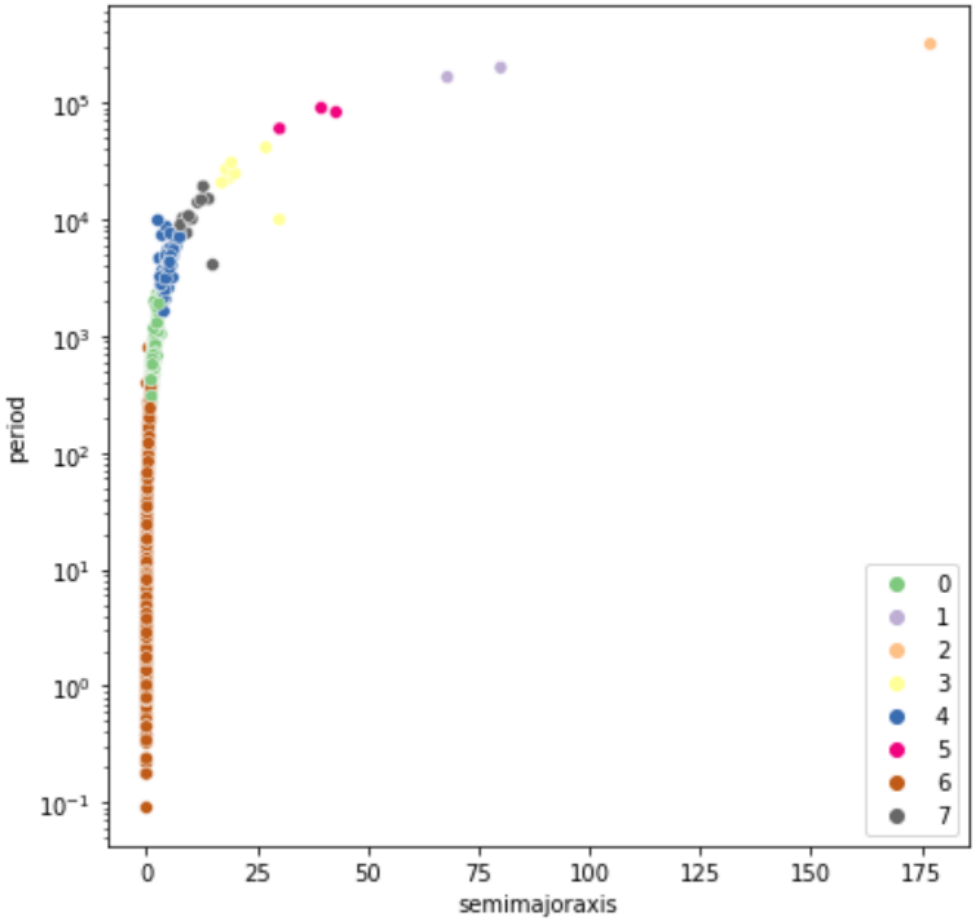
Отримаємо графік:



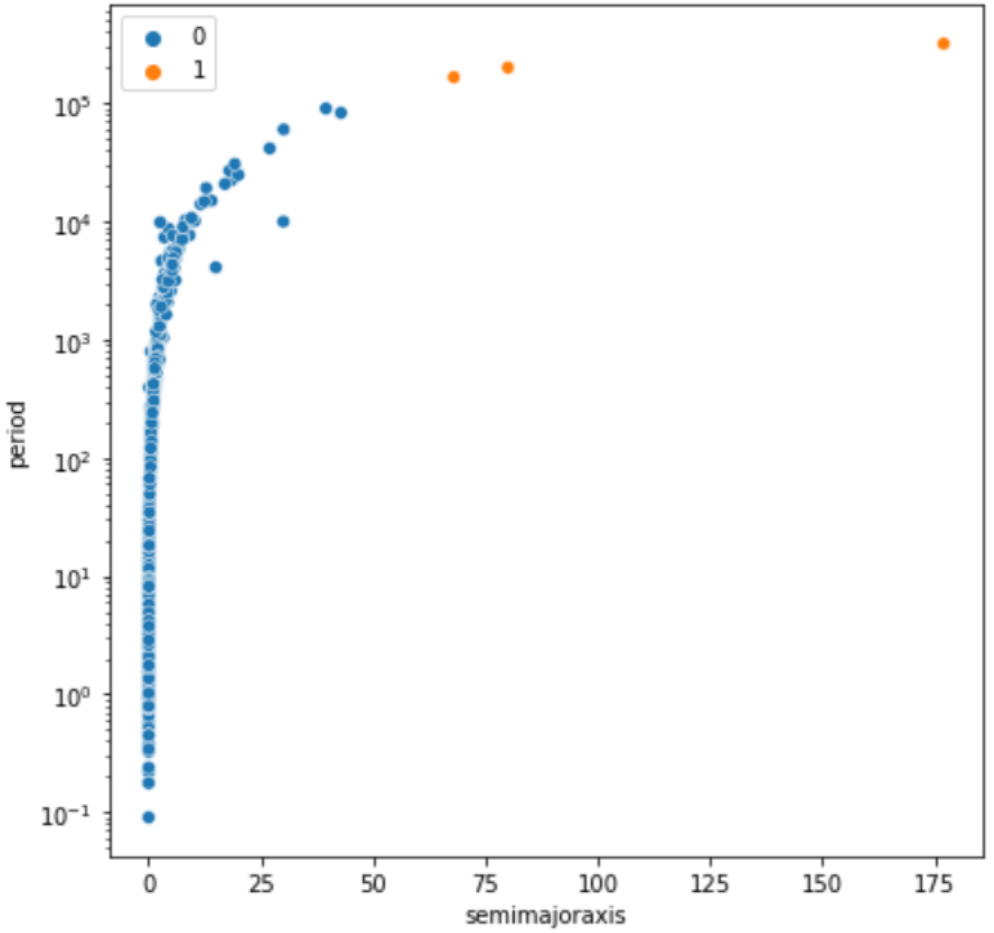
Машинне навчання з scikit-learn

Кластеризація планет:

8 кластерів



2 кластери



Машинне навчання з scikit-learn

Також існують так звані ієрархічні методи кластеризації, зокрема агломеративна кластеризація:

```
from sklearn.cluster import AgglomerativeClustering
ac = AgglomerativeClustering(n_clusters = 8,
linkage='ward').fit(kmeans_data)
```

0.8531116480831661

Машинне навчання з scikit-learn

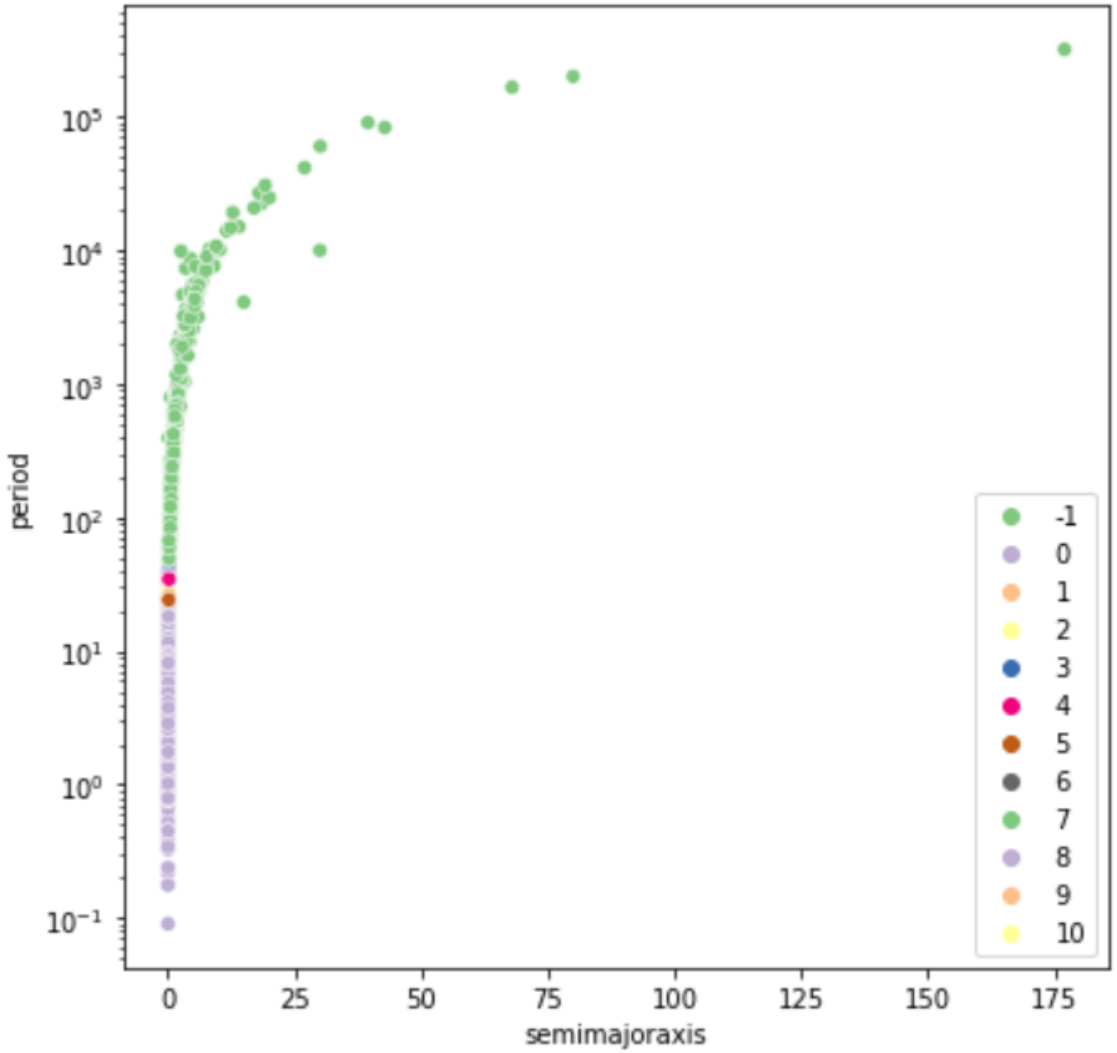
Ще один метод кластеризації не потребує попереднього задання кількості кластерів.

```
from sklearn.cluster import DBSCAN
db = DBSCAN().fit(kmeans_data)
fig, ax = plt.subplots(1, 1, figsize=(7, 7))
ax.set_yscale('log')
sns.scatterplot(x=kmeans_data.semimajoraxis,y=kmeans_data.period,h
ue=db.fit_predict(kmeans_data),palette='Accent',ax=ax)
```

0.08689041861346898

Машинне навчання з scikit-learn

Але він погано працює з кластерами, площини яких не перетинаються:



Машинне навчання з scikit-learn

```
fig, ax = plt.subplots(1, 1, figsize=(5, 5))
sns.scatterplot(x=moons[:, 0],y=moons[:,
1],hue=db.fit_predict(moons),ax=ax)
```