رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثالث

اتخاذ القرار القائم على القواعد

الأنظمة القائمة على القواعد Rule-Based Systems

تُركِّز أنظمة الذكاء الاصطناعي القائمة على القواعد على استخدام مجموعة من القواعد المُحدَّدة مُسبقًا لاتخاذ القرارات وحل المشكلات. **الأنظمة الخبيرة** (Expert Systems) هي المثال الأكثر شهرة للذكاء الاصطناعي القائم على القواعد، وهي إحدى صور الذكاء الاصطناعي الأولى التي طُوِّرت وانتشرت في فترة الثمانينيات والتسعينيات من القرن الماضي. وغالبًا ما كانت تُستخدَم لأتمتة المهام التي تتطلب عادةً خبرات بشرية مثل: تشخيص الحالات الطبية أو تحديد المشكلات التقنية وإصلاحها. واليوم لم تَعُد الأنظمة القائمة على القواعد التقنية هي الأحدث، حيث تفوّقت عليها منهجيات الذكاء الاصطناعي الحديثة. ومع ذلك، لا تزال الأنظمة الخبيرة شائعة الاستخدام في العديد من المجالات نظرًا لقدرتها على الجمع بين الأداء المعقول وعملية اتخاذ القرار البديهية والقابلة للتفسير.

قاعدة المعرفة Knowledge Base

أحد المكونات الرئيسة لأنظمة الذكاء الاصطناعي القائمة على القواعد هي قاعدة المعرِفة، وهي مجموعة من الحقائق والقواعد التي يَستخدِمها النظام لاتخاذ القرارات. تُدخَل هذه الحقائق والقواعد في النظام بواسطة الخبراء البشريين المسؤولين عن تحديد المعلومات الأكثر أهمية وتحديد القواعد التي يتَّبعها النظام. لاتخاذ القرار أو حل المُشكلة، يبدأ النظام الخبير بالتحقق من الحقائق والقواعد في قاعدة البيانات و تطبيقها على الموقف الحالي. إن لم يتمكن النظام من العثور على تطابق بين الحقائق والقواعد في قاعدة المعرِفة، فقد يطلب من المُستخدِم معلومات إضافية أو إحالة المشكلة إلى خبير بشري لمزيد من المساعدة، وإليكَ بعض مزايا وعيوب الأنظمة القائمة على القواعد موضحة في جدول 2.5:

الأنظمة الخبيرة (Expert systems):

النظام الخبير هو أحد أنواع الذكاء الاصطناعي الذي يُحاكي قدرة اتخاذ القرار لدى الخبير البشري. يَستخدِم النظام قاعدة المعرِفة المُكوَّنة من قواعد وحقائق ومحركات الاستدلال لتقديم المشورة أو حل المشكلات في مجال معرفي مُحدَّد.

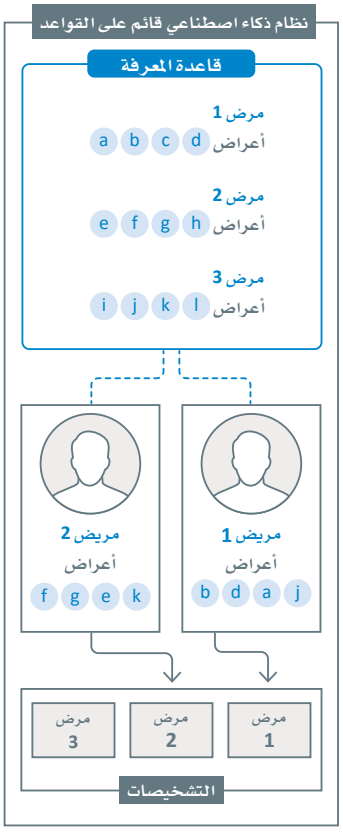
جدول 2.5: المزايا والعيوب الرئيسة للأنظمة القائمة على القواعد

المزايا	العيوب
• يُمكنها اتخاذ القرارات وحل المشكلات بسرعة وبدقة أفضل من البشر، خاصةً عندما يتعلق الأمر بالمهام التي تتطلب قدرًا كبيرًا من المعرِفة أو البيانات. • تَعمل هذه الأنظمة باستمرار، دون تحيُّز أو أخطاء قد تؤثر في بعض الأحيان على اتخاذ القرار البشري.	• تَعمل هذه الأنظمة بكفاءة طالما كانت مُدخَلات المعرِفة والقواعد جيدة، وقد لا تستطيع التعامل مع المواقف التي تقع خارج نطاق خبراتها. • لا يُمكنها التعلُّم أو التكيُّف بالطريقة نفسها مثل البشر، وهذا يجعلها أقل قابلية للتطبيق على الأحداث المُتغيِّرة حيث تتغير مُدخَلات البيانات والمنطق كثيرًا بمرور الوقت.

في هذا الدرس ستتعلّم المزيد حول الأنظمة القائمة على القواعد في سياق أحد تطبيقاتها الرئيسة، وهو: التشخيص الطبي. سيعرض النظام تشخيصًا طبيًا وفقًا للأعراض التي تظهر على المريض، كما هو مُوضّح في الشكل 2.8. بدءًا بنظام تشخيص بسيط مُستنِد إلى القواعد، وستكتشف بعض الأنظمة الأكثر ذكاءً وكيف يُحقّق كل تكرار نتائج أفضل.

الإصدار 1

في الإصدار الأول ستبني نظامًا بسيطًا قائمًا على القواعد يمكنه تشخيص ثلاثة أمراض مُحتملة: KidneyStones (حصى الكُلى)، وAppendicitis (التهاب الزائدة الدودية)، وFood Poisoning (التسمُّم الغذائي). ستكون المُدخَلات إلى النظام هي قاعدة معرفة بسيطة تربط كل مرض بقائمة من الأعراض المُحتملة. يتوفّر ذلك في ملف بتنسيق JSON (جيسون) يُمكنك تحميله وعرضه كما هو مُوضَّح بالأسفل.



شكل 2.8: التشخيص الطبي بواسطة نظام الذكاء الاصطناعي القائم على القواعد

```python
import json # a library used to save and load JSON files

# the file with the symptom mapping
symptom_mapping_file='symptom_mapping_v1.json'

# open the mapping JSON file and load it into a dictionary
with open(symptom_mapping_file) as f:
    mapping=json.load(f)

# print the JSON file
print(json.dumps(mapping, indent=2))
```

```
{
  "diseases": {
    "food poisoning": [
      "vomiting",
      "abdominal pain",
      "diarrhea",
      "fever"
    ],
    "kidney stones": [
      "lower back pain",
      "vomiting",
      "fever"
    ],
    "appendicitis": [
      "abdominal pain",
      "vomiting",
      "fever"
    ]
  }
}
```

سيتَّبع الإصدار الأول القائم على القواعد قاعدة بسيطة ألا وهي: إذا كان لدى المريض على الأقل ثلاثًا من جميع الأعراض المحتملة للمرض، فيجب إضافة المرض كتشخيص مُحتَمل. يمكنك العثور أدناه على دالة Python (البايثون) التي تَستخدِم هذه القاعدة لإجراء التشخيص، بالاستناد إلى قاعدة المعرِفة المذكورة أعلاه وأعراض المرض الظاهرة على المريض.

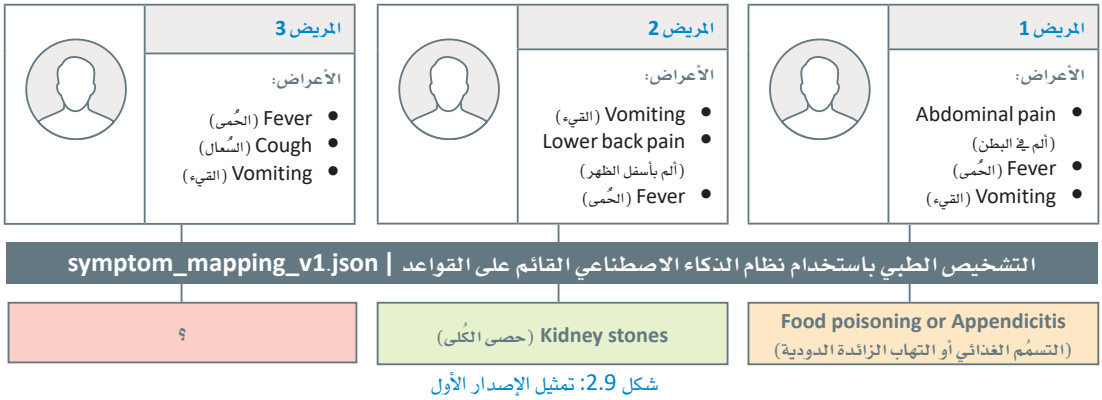
```
def diagnose_v1(patient_symptoms:list):

    diagnosis=[] # the list of possible diseases

    if "vomiting" in patient_symptoms:

        if "abdominal pain" in patient_symptoms:

            if "diarrhea" in patient_symptoms:

                    # 1:vomiting, 2:abdominal pain, 3:diarrhea
                    diagnosis.append('food poisoning')

            elif 'fever' in patient_symptoms:

                    # 1:vomiting, 2:abdominal pain, 3:fever
                    diagnosis.append('food poisoning')
                    diagnosis.append('appendicitis')

        elif "lower back pain" in patient_symptoms and 'fever' in patient_symptoms:

            # 1:vomiting, 2:lower back pain, 3:fever
            diagnosis.append('kidney stones')

    elif "abdominal pain" in  patient_symptoms and\
         "diarrhea" in patient_symptoms and\
         "fever" in patient_symptoms:\
        # 1:abdominal pain, 2:diarrhea, 3:fever
        diagnosis.append('food poisoning')

    return diagnosis
```

في هذه الحالة، تكون قاعدة المعرِفة **محددةً بتعليمات برمجية ثابتة** (Hard-Coded) داخل الدالة في شكل عبارات IF. تَستخدِم هذه العبارات الأعراض الشائعة بين الأمراض الثلاثة للتوصل تدريجيًا إلى التشخيص في أسرع وقت ممكن. على سبيل المثال، عَرض Vomiting (القيء) مشترك بين جميع الأمراض. لذلك، إذا كانت عبارة IF الأولى صحيحة فقد تم بالفعل حساب أحد الأعراض الثلاثة المطلوبة لجميع الأمراض. بعد ذلك، سوف تبدأ في البحث عن Abdominal Pain (ألم البطن) المرتبط بمرضين وتستمر بالطريقة نفسها حتى يتم النظر في جميع مجموعات الأعراض الممكنة.

يُمكنك بعد ذلك اختبار هذه الدالة على ثلاثة مرضى مختلفين:

```
# Patient 1
my_symptoms=['abdominal pain', 'fever', 'vomiting']
diagnosis=diagnose_v1(my_symptoms)
print('Most likely diagnosis:',diagnosis)

# Patient 2
my_symptoms=['vomiting', 'lower back pain', 'fever' ]
diagnosis=diagnose_v1(my_symptoms)
print('Most likely diagnosis:',diagnosis)

# Patient 3
my_symptoms=['fever', 'cough', 'vomiting']
diagnosis=diagnose_v1(my_symptoms)
print('Most likely diagnosis:',diagnosis)
```

```
Most likely diagnosis: ['food poisoning', 'appendicitis']
Most likely diagnosis: ['kidney stones']
Most likely diagnosis: []
```



شكل 2.9: تمثيل الإصدار الأول

يتضمن التشخيص الطبي للمريض الأول التسمُم الغذائي والتهاب الزائدة الدودية لأن الأعراض الثلاثة التي تظهر على المريض ترتبط بكلا المرضين. يُشخَّص المريض الثاني بحصى الكُلى، فهو المرض الوحيد الذي تجتمع فيه الأعراض الثلاثة. في النهاية، لا يُمكن تشخيص الحالة الطبية للمريض الثالث؛ لأن الأعراض الثلاثة التي ظهرت على المريض لا تجتمع في أي من الأمراض الثلاثة.

يتميز الإصدار الأول القائم على القواعد بالبديهية والقابلية للتفسير، كما يتضمن استخدام قاعدة المعرفة والقواعد في التشخيص الطبي دون تَحيُّز أو انحراف عن الخط المعياري. ومع ذلك، يشوب هذا الإصدار العديد من العيوب: أولًا، أن قاعدة ثلاثة أعراض على الأقل هي تمثيل مُبسَّط للغاية لكيفية التشخيص الطبي على يد الخبير البشري. ثانيًا، أن قاعدة المعرفة داخل الدالة تكون محددةً بتعليمات برمجية ثابتة، وعلى الرغم من أنه يسهُل إنشاء عبارات شَرطيَّة بسيطة لقواعد المعرفة الصغيرة، إلا أن المهمة تصبح أكثر تعقيدًا وتستغرق وقتًا طويلًا عند تشخيص الحالات التي تعاني من العديد من الأمراض والأعراض المرضية.

الإصدار 2

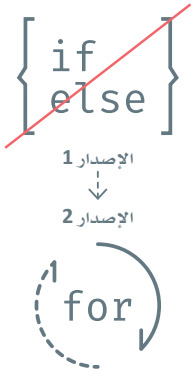
في الإصدار الثاني، ستُعزِّز مرونة وقابلية تطبيق النظام القائم على القواعد بتمكينه من قراءة قاعدة المعرِفة المُتغيِّرة مباشرةً من ملف JSON (جسون). سيؤدي هذا إلى الحد من عملية الهندسة اليدوية لعبارات IF الشَرطيَّة حسب الأعراض ضمن الدالة. وهذا يُعدُّ تحسُّنًا كبيرًا يجعل النظام قابلًا للتطبيق على قواعد المعرِفة الأكبر حجمًا مع تزايد عدد الأمراض والأعراض. وفي الأسفل، مثال يوضّح قاعدة المعرِفة.

```
symptom_mapping_file='symptom_mapping_v2.json'

with open(symptom_mapping_file) as f:
    mapping=json.load(f)

print(json.dumps(mapping, indent=2))
```

```
{
  "diseases": {
    "covid19": [
      "fever",
      "headache",
      "tiredness",
      "sore throat",
      "cough"
    ],
    "common cold": [
      "stuffy nose",
      "runny nose",
      "sneezing",
      "sore throat",
      "cough"
    ],
   "flu": [
      "fever",
      "headache",
      "tiredness",
      "stuffy nose",
      "sneezing",
      "sore throat",
      "cough",
      "runny nose"
    ],
    "allergies": [
      "headache",
      "tiredness",
      "stuffy nose",
      "sneezing",
      "cough",
      "runny nose"
    ]
  }
}
```

قاعدة المعرِفة الجديدة هذه أكبر قليلًا من سابقتها. ومع ذلك، يتَّضح أن محاولة إنشاء عبارات IF الشَرطيَّة في هذه الحالة ستكون أصعب بكثير. على سبيل المثال، تضمنت قاعدة المعرِفة السابقة ربط أحد الأمراض بأربعة أعراض، ومرضين بثلاثة أعراض. وعند تطبيق قاعدة ثلاثة أعراض على الأقل المُطبَّقة في الإصدار الأول، تحصل على 6 مجموعات ثلاثية من الأعراض المحتملة التي تؤخَذ في الاعتبار. في قاعدة المعرِفة الجديدة بالأعلى، تكون للأمراض الأربعة 5 و5 و8 و6 أعراض، على التوالي. وبهذا، تحصل على 96 مجموعة ثلاثية من الأعراض المحتملة. وفي حال التعامل مع مئات أو حتى آلاف الأمراض، ستجدُ أنّه من المستحيل إنشاء نظام مثل الموجود في الإصدار الأول.

وكذلك، لا يوجد سبب طبي وجيه لقِصَر التشخيص الطبي على مجموعات ثلاثية من الأعراض. ولذلك، ستجعل **منطق التشخيص** (Diagnosis Logic) أكثر تنوعًا بحساب عدد الأعراض المُطابقة لكل مرض، والسماح للمُستخدِم بتحديد عدد الأعراض المُطابقة التي يجب توافرها في المرض لتضمينه في التشخيص.



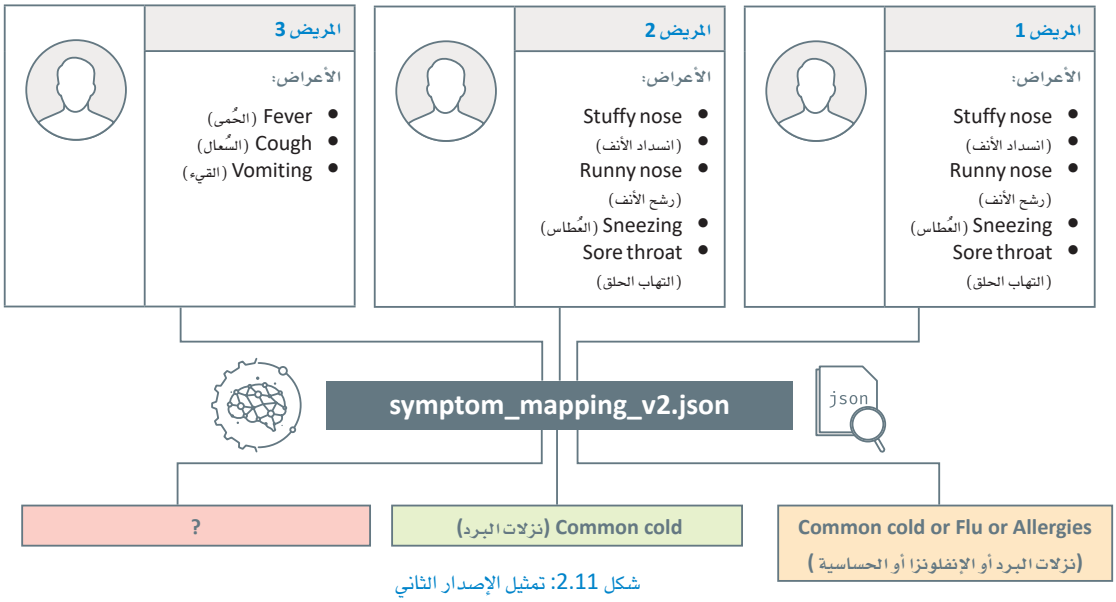
شكل 2.10: الإصدار الثاني لا يحتوي على عبارات IF الشَرطيَّة المحددةً بتعليماتٍ برمجيةٍ ثابتة.

```
def diagnose_v2(patient_symptoms:list,
                symptom_mapping_file:str,
                matching_symptoms_lower_bound:int):

    diagnosis=[]

    with open(symptom_mapping_file) as f:
        mapping=json.load(f)

    # access the disease information
    disease_info=mapping['diseases']

    # for every disease
    for disease in disease_info:

        counter=0

        disease_symptoms=disease_info[disease]

        # for each patient symptom
        for symptom in patient_symptoms:

            # if this symptom is included in the known symptoms for the disease
            if symptom in disease_symptoms:

                counter+=1

        if counter>=matching_symptoms_lower_bound:
            diagnosis.append(disease)

    return diagnosis
```

لا يحتوي هذا الإصدار على عبارات IF الشَرطيَّة المحددة بتعليماتٍ برمجيةٍ ثابتة. بعد تحميل مُخطَّط الأعراض من ملف JSON (جسون)، يبدأ الإصدار في أخذ كلّ مرض محتمل في الاعتبار باستخدام حلقة التكرار الأولى FOR. تتحقق الحلقة من كل عَرَض على حدة بمقارنته بالأعراض المعروفة للمرض وزيادة **العدَّاد** (Counter) في كل مرة يجد فيها النظام تطابقًا.

```
# Patient 1
my_symptoms=["stuffy nose", "runny nose", "sneezing", "sore throat"]
diagnosis=diagnose_v2(my_symptoms,'symptom_mapping_v2.json' , 3)
print('Most likely diagnosis:',diagnosis)

# Patient 2
my_symptoms=["stuffy nose", "runny nose", "sneezing", "sore throat"]
diagnosis=diagnose_v2(my_symptoms, 'symptom_mapping_v2.json' , 4)
print('Most likely diagnosis:',diagnosis)

# Patient 3
my_symptoms=['fever', 'cough', 'vomiting']
diagnosis=diagnose_v2(my_symptoms, 'symptom_mapping_v2.json' , 3)
print('Most likely diagnosis:',diagnosis)
```

```
Most likely diagnosis: ['common cold', 'flu', 'allergies']
Most likely diagnosis: ['common cold']
Most likely diagnosis: []
```



شكل 2.11: تمثيل الإصدار الثاني

لاحظ أن الإصدار الثاني هو نسخة مُعمَّمة من الإصدار الأول. ومع ذلك، يُعدُّ هذا الإصدار أكثر قابلية للتطبيق على نطاق واسع، ويمكن استخدامه كما هو مع أي قاعدة معرفة أخرى بالتنسيق نفسه، حتى لو كانت تشمل الآلاف من الأمراض مع عدد ضخم من الأعراض. كما يَسمح للمُستخدِم بزيادة أو تقليل عدد القيود على التشخيص بضبط المُتغيِّر matching_symptoms_lower_bound. يمكن ملاحظة ذلك في حالة المريض 1 والمريض 2: فعلى الرغم من أنهما يعانيان من الأعراض نفسها، إلا أنه عند ضبط هذا المُتغيِّر، ستحصل على تشخيص مختلف تمامًا.

على الرغم من هذه التحسينات،إلا إنّ بعض العيوب لا تزال موجودة في هذا الإصدار، ولا يُعدُّ تمثيلًا دقيقًا للتشخيص الطبي الحقيقي.

الإصدار 3

في الإصدار الثالث، ستزيد من ذكاء النظام القائم على القواعد بمنحه إمكانية الوصول إلى نوع مُفصَّل من قاعدة المعرِفة. هذا النوع الجديد يأخذ بعين الاعتبار الحقيقة الطبية التي تقول: إنّ بعض الأعراض تكون أكثر شيوعًا من أخرى للمرض نفسه.

```
symptom_mapping_file='symptom_mapping_v3.json'

with open(symptom_mapping_file) as f:
    mapping=json.load(f)

print(json.dumps(mapping, indent=2))
```

```
{
  "diseases": {
    "covid19": {
      "very common": [
        "fever",
        "tiredness",
        "cough"
      ],
      "less common": [
        "headache",
        "sore throat"
      ]
    },
    "common cold": {
      "very common": [
        "stuffy nose",
        "runny nose",
        "sneezing",
        "sore throat"
      ],
      "less common": [
        "cough"
      ]
    },
   "flu": {
      "very common": [
        "fever",
        "headache",
        "tiredness",
        "sore throat",
        "cough"
      ],
      "less common": [
        "stuffy nose",
        "sneezing",
        "runny nose"
      ]
    },
    "allergies": {
      "very common": [
        "stuffy nose",
        "sneezing",
        "runny nose"
      ],
      "less common": [
        "headache",
        "tiredness",
        "cough"
      ]
    }
  }
}
```

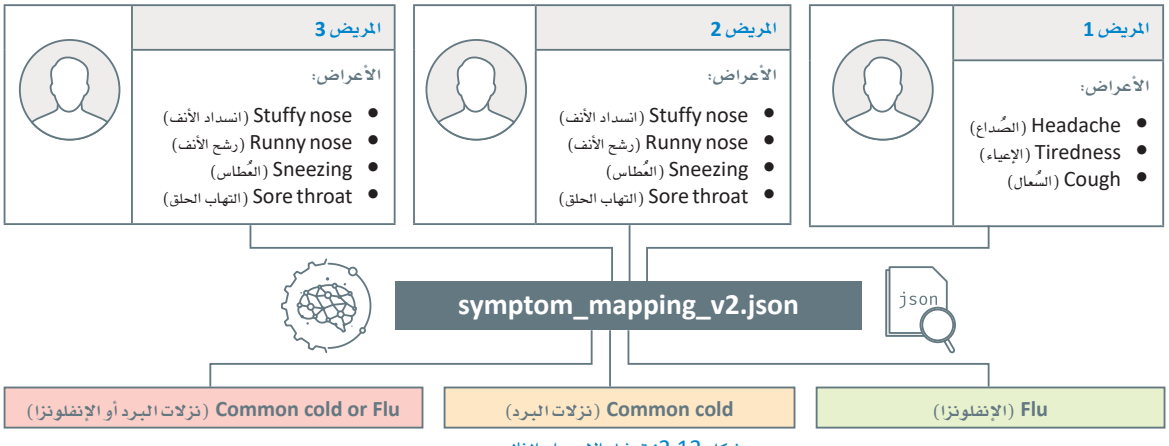
لن يُنظر إلى المنطق الذي يقتصر على عدد الأعراض، وسيُستبدَل بدالة تسجيل النقاط التي تعطي أوزانًا مُخصَّصة للأعراض الأكثر والأقل شيوعًا. ستَتوفر للمستخدِم كذلك المرونة لتحديد الأوزان التي يراها مناسبة. سيتمّ تضمين المرض أو الأمراض ذات المجموع الموزون الأعلى في التشخيص.

```
from collections import defaultdict

def diagnose_v3(patient_symptoms:list,
                symptom_mapping_file:str,
                very_common_weight:float=1,
                less_common_weight:float=0.5
               ):

    with open(symptom_mapping_file) as f:
        mapping=json.load(f)

    disease_info=mapping['diseases']

    # holds a symptom-based score for  each potential disease
    disease_scores=defaultdict(int)

    for disease in disease_info:

        # get the very common symptoms of the disease
        very_common_symptoms=disease_info[disease]['very common']

        # get the less common symptoms for this disease
        less_common_symptoms=disease_info[disease]['less common']

        for symptom in patient_symptoms:

            if symptom in very_common_symptoms:
                disease_scores[disease]+=very_common_weight

            elif symptom in less_common_symptoms:
                disease_scores[disease]+=less_common_weight

    # find the max score all candidate diseases
    max_score=max(disease_scores.values())

    if max_score==0:
        return []

    else:
        # get all diseases that have the max score
        diagnosis=[disease for disease in disease_scores if disease_scores
[disease]==max_score]

        return diagnosis, max_score
```

لكل مرض محتمل في قاعدة المعرفة، تُحدِّد هذه الدالة الجديدة الأعراض الأكثر والأقل ظهورًا على المريض، ثم تزيد من درجة المرض وفقًا للأوزان المُقابِلة، وفي الأخير تُدرَج الأمراض ذات الدرجة الأعلى في التشخيص. يُمكنك الآن اختبار تنفيذ الدالة مع بعض الأمثلة:

```
# Patient 1
my_symptoms=["headache", "tiredness", "cough"]
diagnosis=diagnose_v3(my_symptoms, 'symptom_mapping_v3.json')
print('Most likely diagnosis:',diagnosis)

# Patient 2
my_symptoms=["stuffy nose", "runny nose", "sneezing", "sore throat"]
diagnosis=diagnose_v3(my_symptoms, 'symptom_mapping_v3.json')
print('Most likely diagnosis:',diagnosis)

# Patient 3
my_symptoms=["stuffy nose", "runny nose", "sneezing", "sore throat"]
diagnosis=diagnose_v3(my_symptoms, 'symptom_mapping_v3.json', 1, 1)
print('Most likely diagnosis:',diagnosis)
```

```
Most likely diagnosis: (['flu'], 3)
Most likely diagnosis: (['common cold'], 4)
Most likely diagnosis: (['common cold', 'flu'], 4)
```



شكل 2.12: تمثيل الإصدار الثالث

قد تلاحظ أنه على الرغم من أن الأعراض الثلاثة على المريض 1: Headache (الصداع)، وTiredness (الإعياء)، وCough (السعال) تظهر عند الإصابة بكل من Flu (الإنفلونزا)، و Covid19 (كوفيد- 19). والحساسية، إلّا أنّ الظّاهر في نتائج التّشخيص هي الإنفلونزا فقط. هذا لأن جميع الأعراض الثلاثة شائعة جدًا في قاعدة المعرفة، مما يؤدي إلى درجة قصوى قدرها 3. وبالمثل، في ظل معاناة المريض الثاني والثالث من الأعراض نفسها، تؤدي مُدخَلات الأوزان المختلفة للأعراض الأكثر والأقل شيوعًا إلى تشخيصات مختلفة. وعلى وجه التحديد، يَنتج عن استخدام وزن متساوٍ لنوعين من الأعراض إضافة الإنفلونزا إلى التشخيص.

الإصدار 4

يمكن تحسين النظام القائم على القواعد بزيادة كفاءة قاعدة المعرِفة وتجربة **دوال تسجيل النقاط** (Scoring Functions) المختلفة. وعلى الرغم من أن ذلك سيؤدي إلى تحسين النظام، إلا أنه سيتطلب الكثير من الوقت والجهد اليدوي. ولحسن الحظ، هناك طريقة آلية لبناء نظام مبني على القواعد يكون ذكيًا بما يكفي لتصميم قاعدة معرِفة ودالة تسجيل نقاط خاصة به: باستخدام تعلُّم الآلة. يُطبِّق **تعلُّم الآلة القائم على القواعد** (Rule-Based Machine Learning) خوارزمية تعلّم لتحديد القواعد المُفيدة تلقائيًا، بدلًا من الحاجة إلى الإنسان لتطبيق المعرِفة والخبرات السابقة في المجال لبناء القواعد وتنظيمها يدويًا.

فبدلًا من قاعدة المعرِفة ودالة تسجيل النقاط المُصمَّمتان يدويًا، تَتوقَّع خوارزمية تعلّم الآلة مدخلًا واحدًا فقط وهو مجموعة البيانات التاريخيّة للحالات المَرَضيَّة. فالتعلُّم من البيانات مباشرةً يحوُل دون حدوث المشكلات المرتبطة باكتساب المعرِفة الأساسية والتحقق منها. تتكون كل حالة من بيانات أعراض المريض والتشخيص الطبي الذي يمكن أن يقدمه أي خبير بشري مثل الطبيب. وباستخدام مجموعة بيانات التدريب، تتعلّم الخوارزمية تلقائيًا كيف تتنبأ بالتشخيص المُحتمَل لحالة مريض جديد.

```
import pandas as pd # import pandas to load and process spreadsheet-type data

medical_dataset=pd.read_csv('medical_data.csv') # load a medical dataset.

medical_dataset
```

	fever	cough	tiredness	headache	stuffy nose	runny nose	sneezing	sore throat	diagnosis
0	1	1	1	0	0	0	0	0	covid19
1	0	1	1	1	0	0	0	0	covid19
2	1	1	1	0	0	0	0	0	covid19
3	1	1	1	0	0	0	0	0	covid19
4	1	1	1	0	0	0	0	0	covid19
...	...	...	...	...	...	...	...	...	...
1995	0	1	0	0	1	0	1	1	common cold
1996	0	0	0	1	1	1	1	0	common cold
1997	0	0	1	0	1	0	0	1	common cold
1998	0	0	0	0	1	0	0	1	common cold
1999	0	1	0	0	0	0	1	1	common cold

في المثال أعلاه، تحتوي مجموعة البيانات على 2,000 حالة مرضية، بحيث تتكون كل حالة من 8 أعراض محتملة: Fever (الحُمى)، وCough (السُعال)، وTiredness (الإعياء)، وHeadache (الصُداع)، وStuffy nose (انسداد الأنف)، وRunny nose (رشح الأنف)، وSneezing (العُطاس)، وSore throat (التهاب الحلق). تُرمَّز كل واحدة من هذه الأعراض في عمود ثنائي مُنفصل. العدد الثنائي 1 يشير إلى أن المريض يُعاني من الأعراض، بينما العدد الثنائي 0 يشير إلى أن المريض لا يُعاني من الأعراض.

يحتوي العمود الأخير على تشخيص الخبير البشري، وهناك أربعة تشخيصات محتملة:

Covid19 (كوفيد- 19)، وFlu (الإنفلونزا)، وAllergies (الحساسية)، وCommon cold (نزلات البرد).

يمكنك التحقق من ذلك بسهولة باستخدام المقطع البرمجي التالي بلغة البايثون:

```
set(medical_dataset['diagnosis'])
```

على الرغم من أن هناك العشرات من خوارزميات تعلُّم الآلة المحتملة التي يمكن استخدامها مع مجموعة البيانات هذه، إلا أنك ستستخدم تلك التي تتبع المنهجية المُستنِدة على منطق **شجرة القرار** (Decision Tree)، كما ستَستخدِم DecisionTreeClassifier (مصنّف شجرة القرار) من مكتبة البايثون **سكليرن** (Sklearn) على وجه التحديد.

```
from sklearn.tree import DecisionTreeClassifier

def diagnose_v4(train_dataset:pd.DataFrame):

    # create a DecisionTreeClassifier
    model=DecisionTreeClassifier(random_state=1)

    # drop the diagnosis column to get only the symptoms
    train_patient_symptoms=train_dataset.drop(columns=['diagnosis'])

    # get the diagnosis column, to be used as the classification target
    train_diagnoses=train_dataset['diagnosis']

    # build a decision tree
    model.fit(train_patient_symptoms, train_diagnoses)

    # return the trained model
    return model
```

يُعدُّ تطبيق البايثون في الإصدار الرابع أقصر وأبسط بكثير من التطبيقات السابقة، فهو ببساطة يقرأ الملف التدريبي، ويستخدمه لبناء نموذج شجرة القرار استنادًا إلى العلاقات بين الأعراض والتشخيصات، ومن ثَمَّ ينتج نموذجًا مخصَّصًا. لاختبار هذا الإصدار بشكل صحيح، ابدأ بتقسيم مجموعة البيانات إلى مجموعتين منفصلتين، واحدة للتدريب، وأخرى للاختبار.

```
from sklearn.model_selection import train_test_split

# use the function to split the data, get 30% for testing and 70% for training.
train_data, test_data = train_test_split(medical_dataset, test_size=0.3,
random_state=1)

#print the shapes (rows x columns) of the two datasets
print(train_data.shape)
print(test_data.shape)
```

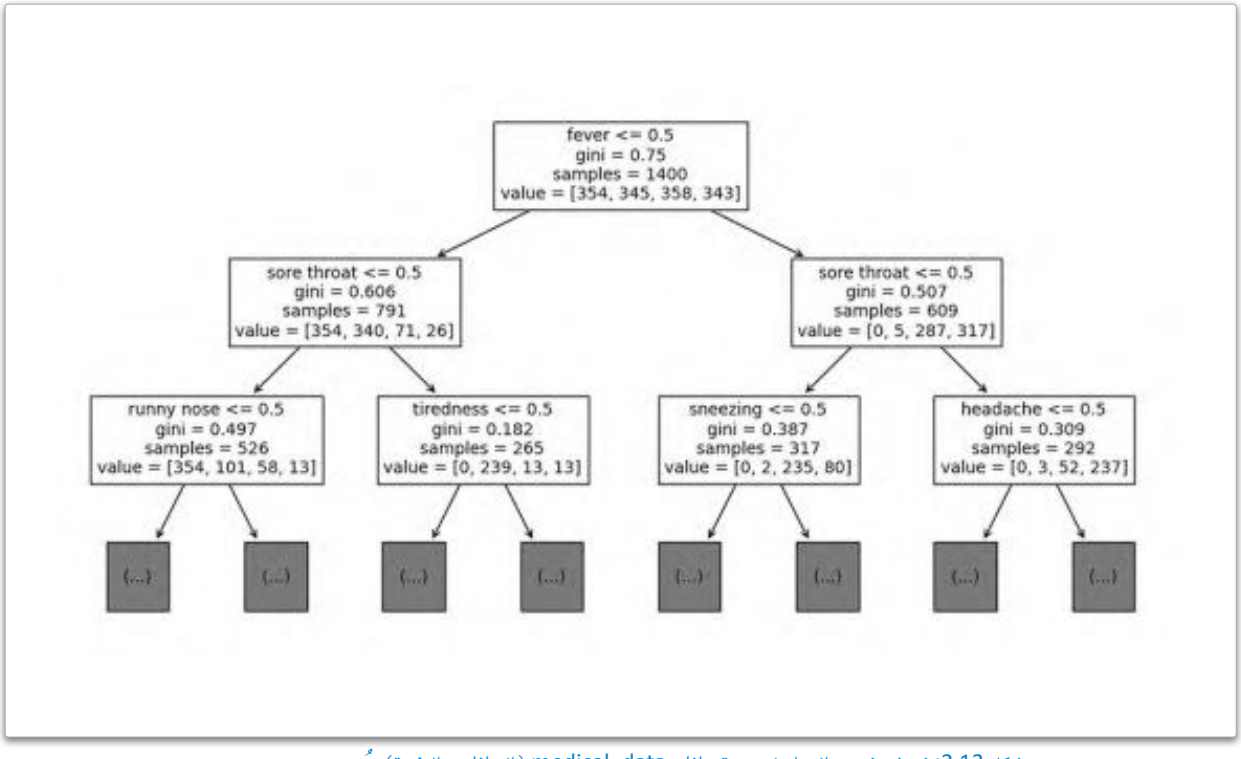
```
(1400, 9)
(600, 9)
```

لديك الآن 1,400 نقطة بيانات ستُستخدَم لتدريب النموذج و600 نقطة ستُستخدَم لاختباره.

ابدأ بتدريب نموذج شجرة القرار وتمثيله:

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

my_tree=diagnose_v4(train_data) # train a model

print(my_tree.classes_) # print the possible target labels (diagnoses)

plt.figure(figsize=(12,6)) # size of the visualization, in inches

# plot the tree
plot_tree(my_tree,
          max_depth=2,
          fontsize=10,
          feature_names=medical_dataset.columns[:-1]
         )
```

```
['allergies' 'common cold' 'covid19' 'flu']
```



شكل 2.13: نموذج شجرة القرار لمجموعة بيانات medical_data (البيانات- الطبية) بعُمق مستويين

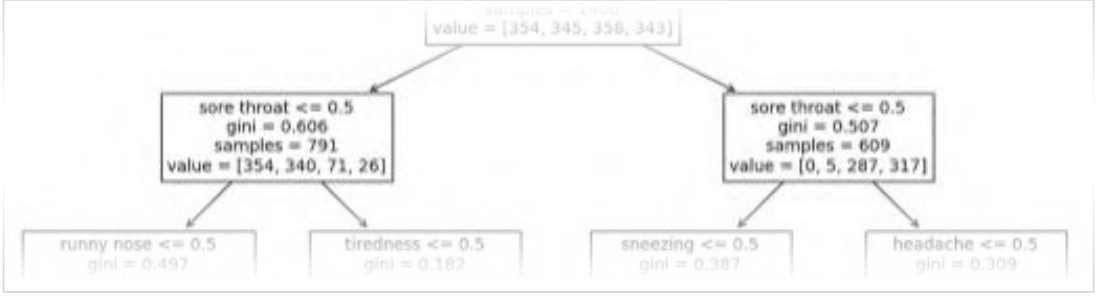
تُستخدَم دالة ( )plot_tree لرسم وعرض شجرة القرار. ولعدم توفر مساحة كافية للعرض سيتم تمثيل المستويين الأوّلين فقط، بالإضافة إلى الجذر. يمكن ضبط هذا الرقم بسهولة باستخدام المُتغيِّر max_depth.

```
# plot the tree
plot_tree(my_tree,
          max_depth=2,
          fontsize=10
```

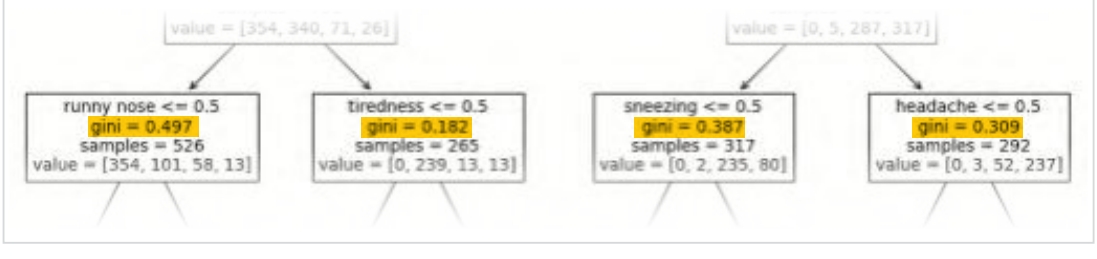
عُمق شجرة القرار.

كل عُقدة في الشجرة تُمثِّل مجموعة فرعية من المرضى، فعلى سبيل المثال،

تُمثِّل عُقدة الجذر إجمالي عدد 1,400 مريض في مجموعة بيانات التدريب. من بينهم، 354، و345، و358، و343 شُخِّصوا بـ Allergies (الحساسية)، وCommon cold (نزلات البرد)، وCovid19 (كوفيد-19)، وFlu (الإنفلونزا)، على التوالي.



بُنيَّت الشجرة باستخدام نمط من الأعلى إلى الأسفل عبر **التفرُّع الثنائي** (Binary Splits). يَستنِد التفرُّع الأول إلى ما إذا كان المريض يُعاني من الحُمى أم لا. ونظرًا لأن كل خصائص الأعراض ثنائية، يكون التحقق a <=0.5 صحيحًا إذا لم يكن المريض يعاني من الأعراض. أما المرضى الذين لا يعانون من الحُمى (المسار الأيسر) يتفرَّعون مرة أخرى بناءً على ما إذا كانوا يعانون من التهاب الحلق أم لا. المرضى الذين لا يعانون من التهاب الحلق يتفرَّعون بناءً على ما إذا كانوا يعانون من رشح الأنف أم لا. في هذه المرحلة، تحتوي العُقدة على 526 حالة. تمّ تشخيص 354، و101، و58، و13 من بينهم بالحساسية، ونزلات البرد، وكوفيد-19، والإنفلونزا، على التوالي.



يستمر التفرُّع حتى تُحدِّد الخوارزمية الحالات التي انقسمت بالفعل إلى عُقد نقيَّة تمامًا. العُقدة النقيَّة بالكامل تحتوي على الحالات التي لها التشخيص نفسه. قِيَم مؤشر gini (جيني) المُحدَّدة على كل عُقدة، تُمثِّل مؤشرات على مقياس جيني، وهي صيغة شهيرة تُستخدَم لتقييم درجة نقاء العُقدة.

يقيس مؤشر جيني (Gini Index) الشوائب بالعُقدة، وبالتحديد احتمالية تصنيف محتويات العُقدة بصورة خاطئة. يشير انخفاض مُعامِل جيني إلى ارتفاع درجة تأكُّد الخوارزمية من التصنيف.

ستَستخدِم الآن شجرة القرار للتنبؤ بالتشخيص الأكثر احتمالًا للمرضى في مجموعة الاختبار.

تُستخدَم مجموعة الاختبار لتقييم أداء النموذج. تَستنِد طريقة التقييم الدقيقة على ما إذا كان المقصود من المهمة **الانحدار** (Regression) أم **التصنيف** (Classification). في مثل مشكلات التصنيف المعروضة هنا، تُستخدَم طرائق التقييم الشهيرة مثل: حساب **دقة النموذج** (Model's Accuracy) و**مصفوفة الدقة** (Confusion Matrix).

- الدقة هي نسبة التنبؤات الصحيحة التي يقوم بها المُصنِّف. تَحقُّق دقة عالية قريبة من 100% يعني أن معظم التنبؤات التي يقوم بها المُصنِّف صحيحة.
- مصفوفة الدقة هي جدول يقارن بين القيم الحقيقية (الفعلية) وبين التنبؤات التي يقوم بها المُصنِّف في مجموعة البيانات. يحتوي الجدول على صف واحد لكل قيمة صحيحة وعمود واحد لكل قيمة مُتوقَّعة. كل مُدخَل في المصفوفة يُمثِّل عدد الحالات التي لها قيم فعلية ومُتوقَّعة.

```
# functions used to evaluate a classifier
from sklearn.metrics import accuracy_score,confusion_matrix

# drop the diagnosis column to get only the symptoms
test_patient_symptoms=test_data.drop(columns=['diagnosis'])

# get the diagnosis column, to be used as the classification target
test_diagnoses=test_data['diagnosis']

# guess the most likely diagnoses
pred=my_tree.predict(test_patient_symptoms)

# print the achieved accuracy score
accuracy_score(test_diagnoses,pred)
```

```
0.8166666666666667
```

ستلاحِظ أن شجرة القرار تُحقِّق دقة تصل إلى 81.6%، وهذا يعني أنه من بين 600 حالة تمّ اختبارها، شَخَّصت الشجرة 490 منها بشكل صحيح. يُمكنك كذلك طباعة مصفوفة الدقة للنموذج لتستعرض بشكل أفضل الأمثلة المُصنَّفة بشكل خاطئ.

```
confusion_matrix(test_diagnoses,pred)
```

```
array([[143,   3,   0,   0],
       [ 48,  98,   5,   4],
       [  2,   1, 127,  12],
       [  1,   3,  31, 122]])
```

	الحساسية المُتوقَّعة	نزلات البرد المُتوقَّعة	كوفيد19- المُتوقَّع	الإنفلونزا المُتوقَّعة
الحساسية الفعلية	**143**	3	0	0
نزلات البرد الفعلية	48	**98**	5	4
كوفيد-19 الفعلي	2	1	**127**	12
الإنفلونزا الفعلية	1	3	31	**122**

شكل 2.14: مصفوفة الدقة للحالات المُتوقَّعة والحالات الفعلية

الأرقام الواقعة في الخط القُطري (المُظللة باللون الوردي) تُمثِّل الحالات المُتوقَّعة بشكل صحيح، أما الأرقام التي تقع خارج الخط القُطري فتُمثِّل أخطاء النموذج.

على سبيل المثال، بالنظر إلى ترتيب التشخيصات الأربعة المُحتملة [Allergies (الحساسية)، Common cold (نزلات البرد)، Covid19 (كوفيد-19)، Flu (الإنفلونزا)]، توضح المصفوفة أن النموذج أخطأ في تصنيف 48 حالة من المُصابين بنزلات البرد بأنهم مصابون بالحساسية، كما أخطأ في تصنيف 31 حالة من المُصابين بالإنفلونزا بأنهم مصابون بكوفيد-19.

وعلى الرغم من أنّ هذا النموذج ليس مثاليًا، فمن المُثير للدهشة أنّه قادر على تحقيق مثل هذه الدرجة العالية من الدقة بتعلُّم مجموعة القواعد الخاصة به، دون الحاجة إلى قاعدة معرفة أنشئت يدويًا. بالإضافة إلى تحقيق مثل هذه الدقة دون محاولة ضبط مُتغيرات الأداء المتنوعة لـ DecisionTreeClassifier (مُصنِّف شجرة القرار). وبالتالي، يُمكن تحسين دقة النموذج لأفضل من ذلك. كما يُمكن تحسين النموذج بتجاوز قيود النموذج القائم على القواعد وتجربة أنواع مختلفة من خوارزميات تعلُّم الآلة. وستتعلّم بعض هذه الطرائق في الوحدة التالية.

تمرينات

1 اذكر بعض مزايا وعيوب الأنظمة القائمة على القواعد.

2 ما مزايا وعيوب الإصدار الأول؟

3 أضف إلى المقطع البرمجي الخاص بالإصدار الأول لنظام قائم على القواعد مريضًا يُعاني من الأعراض التالية [Vomiting (القيء)، وAbdominal pain (آلام البطن)، وDiarrhea (الإسهال)، وFever (الحُمى)، وLower back pain (ألم بأسفل الظهر)]. ما التشخيص الطبي لحالة المريض؟ دَوِّن ملاحظاتك بالأسفل.