



وظيفة مقرر الخوارزميات الذكية BIA601

الدكتور: عصام محمد سليمان

إعداد الطالب:

abdallah_152063	Templates
mohammad_yaman_156251	algorithms / genetic_feature_selection.py
asmaa_199515	Utilis
saleh_203178	algorithms/ __pycache__ + algorithms / __init__.py
mohammad_213142	HOW_TO_RUN_FLASK.md+ Read.md+ uploads
abdoulrahman_204246	app.py+ requirements.txt
salwa_216113	تقرير
yosef_113590	algorithms/traditional_methods.py

رابط الموقع:

/https://bia601hw.pythonanywhere.com

:GitHub رابط

https://github.com/Abdulrahman1204/App_BIA601.git

نظرة عامة على المشروع

الهدف الرئيسي:

المشروع هو تطبيق ويب يستخدم الخوارزمية الجينية (Genetic Algorithm) لاختيار أفضل مجموعة من الميزات (Features) من بين آلاف الميزات في مجموعة البيانات.

الأهداف:

- تقليل عدد الميزات المستخدمة
- الحفاظ على دقة عالية في التصنيف
- تسريع عملية التدريب
- تقليل احتمالية Overfitting

الميزات الرئيسية:

- تطبيق الخوارزمية الجينية لاختيار الميزات بشكل ذكي
- عرض تقدم العمل في الوقت الفعلي (Real-time)
- مقارنة النتائج مع ٤ طرق تقليدية معروفة
- واجهة مستخدم جميلة وسهلة الاستخدام
- إعدادات سريعة لاختبار السريع

• الشكلة التي يحلها المشروع

الشكلة الأساسية:

عند العمل علىمجموعات بيانات كبيرة (Big Data)، نواجه مشاكل عديدة:

حجم البيانات الكبير:

مثال: بيانات تحتوي على ١٠٠٠ صفات و ٥٠٠ ميزة

- سنحتاج لحوالي ٥٠٠,٠٠٠ قيمة للتدريب

- كل ميزة إضافية تزيد التعقيد بشكل كبير

التحديات الرئيسية:

١) التحدي: وقت التدريب الطويل

الشرح: كل ميزة إضافية تزيد وقت التدريب

المثال: $500 \text{ ميزة} = 5 \text{ ساعات}$, $100 \text{ ميزة} = 30 \text{ دقيقة}$

٢) التحدي: Overfitting

الشرح: النموذج يحفظ البيانات بدلاً من التعلم

المثال: دقة ٩٩٪ في التدريب، ٦٠٪ في الاختبار

٣) التحدي: صعوبة الاختيار

الشرح: أي ٥٠ ميزة يجب اختيارها من ٥٠٠

المثال: عدد الاحتمالات: $(1500 / (150 \times 1450))!$

٤) التحدي: التفسير الصعب

الشرح: كيف نفس نموذج بـ ٥٠٠ ميزة؟

المثال: صعب جداً فهم قرارات النموذج

الحل المقترن:

استخدام الخوارزمية الجينية لاختيار تلقائي ذكي:

١. تجربة تركيبات مختلفة من الميزات
٢. تقييم كل تركيبة حسب الأداء
٣. تحسين الحل تدريجياً عبر الأجيال
٤. الوصول لأفضل تركيبة موازنة بين الدقة والعدد

• الخوارزمية الجينية (Genetic Algorithm)

ما هي الخوارزمية الجينية؟

الخوارزمية الجينية هي نوع من الخوارزميات التطورية التي تحاكي عملية التطور البيولوجي في الطبيعة.

التشبيه البسيط:

الكائنات الحية تتطور عبر الأجيال



الクロموسومات (Chromosomes) تمثل الحلول



الأفضل يبقى والأضعف يموت



التهجين والطفرات تحسن الحل

كيف تعمل في مشروعنا:

المرحلة ١: التمثيل (Representation)

الクロموسوم (Chromosome): يمثل مجموعة من الميزات المختارة

مثال: بيانات تحتوي على ١٠٠ ميزة

الクロموسوم: [..., ٠, ١, ٠, ١, ٠, ٠, ١, ٠, ٠, ١, ...]

| | | | | | | |

ميزة ١ ميزة ٢ ميزة ٣ ميزة ٤ ...

٠ = الميزة غير مستخدمة

١ = الميزة مستخدمة

في الكود:

```
def create_chromosome(self):  
    n_features = self.X.shape[1]  
    # اختيار ٣٠٪ من الميزات عشوائياً  
    n_selected = random.randint(int(n_features * 0.3), int(n_features * 0.7))  
    selected = random.sample(range(n_features), n_selected)  
    chromosome = [0] * n_features  
    for idx in selected:  
        chromosome[idx] = 1 # تفعيل الميزة  
    return chromosome
```

المرحلة ٢: حساب اللياقة (Fitness)

اللياقة: تقييم جودة كل كروموسوم (حل)

اللياقة = دقة التصنيف - معاقبة عدد الميزات الكبير

:مثال

- كروموسوم ١: دقة٪٨٥، ٤٠ ميزة → لياقة٪٨٥ - ٪٤ = ٪٨١
- كروموسوم ٢: دقة٪٨٢، ٢٠ ميزة → لياقة٪٨٢ - ٪٢ = ٪٨٠

كروموسوم ١ أفضل!

في الكود:

```
def fitness(self, chromosome):  
    selected_features = [i for i, bit in enumerate(chromosome) if bit == 1]  
  
    if len(selected_features) == 0:  
        return 0  
  
    X_selected = self.X_scaled[:, selected_features]  
    model = RandomForestClassifier(n_estimators=20, random_state=42, n_jobs=-1)  
  
    scores = cross_val_score(model, X_selected, self.y, cv=5, scoring='accuracy', n_jobs=-1)  
    accuracy = scores.mean()
```

```
# معاقبة استخدام عدد كبير من الميزات
```

```
penalty = len(selected_features) / self.X.shape[1] * 0.1

return accuracy - penalty
```

المرحلة ٣: الانتخاب (Selection)

اختيار أفضل الكائنات للتكرار

الجيل الحالي: ٣٠ كروموسوم



حساب اللياقة لكل واحد



اختيار الأفضل للجيل القادم

في الكود:

```
def select_parents(self, population, fitness_scores):

    max_fitness = max(fitness_scores)

    fitness_scores = [f + abs(min(fitness_scores)) + 1 for f in fitness_scores]

    total = sum(fitness_scores)

    probabilities = [f / total for f in fitness_scores]

    parent1 = np.random.choice(len(population), p=probabilities)

    parent2 = np.random.choice(len(population), p=probabilities)

    return population[parent1], population[parent2]
```

المرحلة ٤: التهجين (Crossover)

دمج صفات الآب و الأم لإنشاء أبناء أفضل

الأب ١: [١, ٠, ١, ٠, ٠ | ١, ١, ٠, ٠, ١]

الأب ٢: [٠, ١, ٠, ١, ١ | ٠, ٠, ١, ١, ٠]

الابن ١: [٠, ١, ٠, ١, ١ | ١, ١, ٠, ١, ٠]

الابن ٢: [١, ٠, ١, ٠, ٠ | ٠, ٠, ١, ١, ٠]

في الكود:

```
def crossover(self, parent1, parent2):  
    if random.random() > self.crossover_rate:  
        return parent1.copy(), parent2.copy()  
  
    point = random.randint(1, len(parent1) - 1)  
    child1 = parent1[:point] + parent2[point:]  
    child2 = parent2[:point] + parent1[point:]  
    return child1, child2
```

المرحلة ٥: الطفرات (Mutation)

إدخال تغييرات عشوائية صغيرة

قبل الطفرة: [١, ٠, ٠, ١, ١, ٠, ١]

↓

بعد الطفرة: [٠, ١, ١, ١, ٠, ٠, ١] (تغيير في الميزة ٣)

في الكود:

```
def mutate(self, chromosome):  
    for i in range(len(chromosome)):  
        if random.random() < self.mutation_rate:  
            chromosome[i] = 1 - chromosome[i] # تبديل . ↔ ١  
    return chromosome
```

دورة الحياة الكاملة

:def evolve(self)

```
❶ إنشاء الجيل الأول  
population = [self.create_chromosome() for _ in range(self.population_size)]  
  
❷ حساب اللياقة  
for generation in range(self.generations):  
    fitness_scores = [self.fitness(chrom) for chrom in population]
```

٣٦# حفظ الأفضل

```
best_chromosome = population[np.argmax(fitness_scores)]
```

٤٧# بث التحديثات في الوقت الفعلي

```
if self.callback:  
    self.callback}  
  
'    generation': generation + 1,  
'    progress': (generation + 1) / self.generations * 100,  
'    best_fitness': max(fitness_scores),  
'    avg_fitness': np.mean(fitness_scores),  
'    n_features': sum(best_chromosome)  
{
```

٥٨# إنشاء الجيل الجديد

```
new_population = [best_chromosome.copy()]
```

```
while len(new_population) < self.population_size:  
    parent1, parent2 = self.select_parents(population, fitness_scores)  
    child1, child2 = self.crossover(parent1, parent2)  
    child1 = self.mutate(child1)  
    child2 = self.mutate(child2)  
    new_population.extend([child1, child2])
```

```
population = new_population[:self.population_size]
```

```
return best_chromosome
```

الإعدادات القابلة للتغيير

:Population Size

القيمة الافتراضية: ٣٠

الشرح: عدد الحلول في كل جيل

التأثير: كلما زاد → أفضل نتائج لكن أبطأ

:Generations

القيمة الافتراضية: ٢٠

الشرح: عدد الأجيال

التأثير: كلما زاد → أفضل نتائج لكن أبطأ

:Mutation Rate

القيمة الافتراضية: ٠,١

الشرح: احتمالية الطفرة

التأثير: زيادة → تنوع أكثر

:Crossover Rate

القيمة الافتراضية: ٠,٨

الشرح: احتمالية التهجين

التأثير: زيادة → تجربة أكثر

• هيكل الموقع والتطبيق

التقنيات المستخدمة

:الواجهة الأمامية (Frontend)

- HTML5 - هيكل الصفحة

- CSS3 - التصميم والأنميات

- JavaScript - التفاعل والتحديثات في الوقت الفعلي

الخلفية (Backend):

- إطار عمل Python للويب Flask
- معالجة البيانات Pandas
- العمليات الرياضية NumPy
- Machine Learning - خوارزميات Scikit-learn

التخزين المؤقت:

- في الذاكرة (In-memory) - للجلسات

• التحديثات في الوقت الفعلي (Real-time Updates)

المشكلة السابقة:

قبل التحديث:

المستخدم يرفع الملف



(انتظار طويل - لا يعرف ماذا يحدث!)



ظهور النتائج فجأة

المشكلة المستخدم لا يعرف:

- هل التطبيق عالق؟
- كم تبقى من الوقت؟
- أين وصلنا في المعالجة؟
- هل تحدث أخطاء؟

الحل: Server-Sent Events (SSE)

SSE تسمح للسيرفر بإرسال تحديثات للمتصفح في الوقت الفعلي!

المستخدم يرفع الملف



اتصال SSE



تحديثات كل جيل:

— الجيل ٢٠/١ (%) ٥٪

— الجيل ٢٠/٢ (%) ١٠٪

— الجيل ٢٠/٣ (%) ١٥٪

... — L



انتهاء المعالجة

• كيفية استخدام الموقع

الدخول الى رابط الموقع:

<https://bia601hw.pythonanywhere.com>

اختيار الميزات باستخدام الخوارزمية الجينية

استخدم الخوارزميات الذكية لإيجاد أفضل مجموعة من الميزات

معلومات عن المشروع

الهدف من المشروع

يهدف هذا المشروع إلى تطوير قدرات الطالب على حل مشكلة عملية باستخدام الخوارزميات الذكية، عند العمل علىمجموعات بيانات ضخمة تحتوي على المئات أو الآلاف من الميزات، تواجه تحديات كبيرة:

Overfitting

احتمال حدوث Overfitting عند استخدام ميزات كثيرة

وقت التدريب

زيادة وقت التدريب للنماذج بشكل كبير

صعوبة الاختبار

صعوبة في اختيار الميزات الأكثر أهمية من بين المئات

التفسير

صعوبة تفسير النماذج المعقدة

الحل المقترن

استخدام الخوارزمية الجينية (**Genetic Algorithm**) للعثور على مجموعة الميزات المثلث التي تعطي أداءً أفضل للنموذج مع تقليل عدد الميزات المستخدمة بشكل تلقائي.

الطرق المستخدمة للمقارنة

✓ **الخوارزمية الجينية** (Genetic Algorithm)

✓ **F-Test** (Statistical Method)

✓ **Mutual Information** (Information Theory)

✓ **RFE** (Recursive Feature Elimination)

✓ **Model-Based** (Random Forest Feature Importance)

ابداً التحليل

رفع ملف البيانات

قم بسحب وإفلات ملف CSV أو اضغط للتصفح

اختر ملف CSV

اختر عمود الهدف (Target):

— اختر عمود الهدف —

معدل الطرفatas:

عدد الأجيال:

حجم المجتمع:

٠.٣

٢٠

٢٠

ابداً التحليل

تجربة رفع ملف للبيانات:

اختيار الميزات باستخدام الخوارزمية الجينية

استخدم الخوارزميات الذكية لإيجاد أفضل مجموعة من الميزات

معلومات عن المشروع

الهدف من المشروع
يهدف هذا المشروع إلى تطوير قدرات الطالب على حل مشكلة عملية باستخدام الخوارزميات الذكية. عند العمل علىمجموعات بيانات تضم مئات أو الآلاف من الميزات، تواجه تحديات كبيرة:

- Overfitting**: احتمال حدوث Overfitting عند استخدام ميزات كثيرة.
- وقت التدريب**: زيادة وقت التدريب للنموذج بشكل كبير.
- صعوبة الاختبار**: صعوبة في اختيار الميزات الأكثر أهمية من بين المئات.
- التفسير**: صعوبة تفسير النماذج المعقدة.

الحل المقترن

استخدام الخوارزمية الجينية (**Genetic Algorithm**) للعثور على مجموعة الميزات المثلث التي تعطي أفضل أداء للنموذج مع تقليل عدد الميزات المستخدمة بشكل تلقائي.

الطرق المستخدمة للمقارنة

- ✓ **الخوارزمية الجينية** (Genetic Algorithm)
- ✓ **F-Test** (Statistical Method)
- ✓ **Mutual Information** (Information Theory)
- ✓ **RFE** (Recursive Feature Elimination)
- ✓ **Model-Based** (Random Forest Feature Importance)

ابدا التحليل

رفع ملف البيانات
قم بسحب وإفلات ملف CSV أو اضغط للتصفح
اختر ملف CSV

الملف: iris.csv | الحجم: KB 2.86 | الصفوف: 150 | الأعمدة: 5

اختر عمود الهدف (Target): target

معدل الطرفات: ٥٠

عدد الاجيال: ٢٠

حجم المجتمع: ٣٠

ابدا التحليل

بدأ بالعمل:

اختيار الميزات باستخدام الخوارزمية الجينية

استخدم الخوارزميات الذكية لإيجاد أفضل مجموعة من الميزات

معلومات عن المشروع

الهدف من المشروع

يهدف هذا المشروع إلى تطوير خدمة الطالب على حل مشكلة عملية باستخدام الخوارزميات الذكية. عند العمل علىمجموعات بيانات ضخمة تحتوي على المئات أو الآلاف من الميزات، تواجه تحديات كبيرة:

Overfitting

احتمال حدوث Overfitting عند استخدام ميزات كثيرة

وقت التدريب

زيادة وقت التدريب للنموذج بشكل كبير

صعوبة الاختبار

صعوبة في اختيار الميزات الأكثر أهمية من بين المئات

القصور

صعوبة تفسير النماذج المعقدة

الحل المقترن

استخدام الخوارزمية الجينية (Genetic Algorithm) للعنور على مجموعة الميزات المثلث التي تعطي أفضل أداء للنموذج مع تقليل عدد الميزات المستخدمة بشكل تلقائي.

طرق المستخدمة للمقارنة

الخوارزمية الجينية (Genetic Algorithm)

F-Test (Statistical Method)

Mutual Information (Information Theory)

RFE (Recursive Feature Elimination)

Model-Based (Random Forest Feature Importance)

ابداً التحليل

رفع ملف البيانات

قم بسحب وإفلات ملف CSV أو اضغط للتصفح

اختر ملف CSV

الملف: iris.csv | الحجم: 2.86 KB | الصفوف: 150 | الأعمدة: 5

اختر عمود الهدف (Target):

target

معدل الظفرات:

٠.٦

عدد الأجيال:

٢٠

حجم المجتمع:

٣٠

ابداً تحليل

تقدم المعالجة

جارى التحليل ...

انتظار النتيجة:

اختيار الميزات باستخدام الخوارزمية الجينية

استخدم الخوارزميات الذكية لإيجاد أفضل مجموعة من الميزات

معلومات عن المشروع

الهدف من المشروع

يهدف هذا المشروع إلى تطوير قدرات الطالب على حل مشكلة عملية باستخدام الخوارزميات الذكية. عند العمل على مجموعة بيانات ضخمة تحتوي على الملايين أو الآلاف من الميزات، تواجه صعوبات كبيرة:

- Overfitting (التجزء) - احتمال حدوث Overfitting عند استخدام ميزات كثيرة
- زيادة وقت التدريب: التمدد بشكل كبير
- صعوبة الاحتدام (Overfitting) في اختيار الميزات الأكثر أهمية من بين الملايين
- صعوبة تفسير النماذج السعيدة

الحل المقترن

استخدام الخوارزمية الجينية (Genetic Algorithm) للعنور على مجموعة الميزات المتلائمة التي تعطي أداءً للنموذج مع تقليل عدد الميزات المستخدمة بشكل تلقائي.

الطرق المستخدمة للمقارنة

- الخوارزمية الجينية (Genetic Algorithm)
- F-Test (Statistical Method)
- Mutual Information (Information Theory)
- RFE (Recursive Feature Elimination)
- Model-Based (Random Forest Feature Importance)

ابدا التحليل

رفع ملف البيانات

قم بسحب وافلات ملف CSV أو اضغط هنا لتصفح المتصفح

CSV

الملف: iris.csv | المحجم: 2.06 KB | المصفوفة: 150 | الأعمدة: 5

اختر عمود الهدف (Target): target

عدد الميزات: 4

عدد الأجيال: 20

خطيم المحظوظ: 20

ابدا التحليل

النتائج

أجمالي الميزات	4	تقليل الميزات	75.0%	عدد الميزات المستهارة	1	الدقة	92.83%
----------------	---	---------------	-------	-----------------------	---	-------	--------

المقارنة مع الطرق التقليدية

الطريقة	الدقة
الخوارزمية الجينية	92.83%
F-Test	92.00%
Mutual Information	92.00%
RFE	92.00%
Model-Based	95.33%

الميزات المختاراة

الميزات المختاراة بواسطة الخوارزمية الجينية (1 ميزة): petal width (cm)

ملاحظة مهمة: ان الظهور الفجائي للنتيجة والمتاخر أيضا هو بسبب الاستضافة المجانية اما عندما نقوم بتجربة المشروع محليات ف يتم اخطار المستخدم بما يتم عمله فعليا على السيرفر والمرحلة الحالية من

العمل

• الخلاصة

ما تعلمناه

المشكلة: مجموعات البيانات الكبيرة بها ميزات كثيرة
الحل: استخدام الخوارزمية الجينية لاختيار الذكي
المقارنة: ٥ طرق مختلفة، كل واحدة مناسبة لحالة
التحديات: بث مباشر للتقدم
التصميم: واجهة جميلة وسهلة

الاستخدامات المستقبلية

الملاحظات:

- الطب الحيوي (اختيار الجينات المهمة)
- الرياضة (اختيار المؤشرات)
- التسويق (اختيار المتغيرات)
- المالية (اختيار المؤشرات)

التحسينات المقترحة

أكمل:

- رسم بياني للتطور عبر الأجيال
- حفظ النتائج وتاريخ التشغيل
- خيارات متقدمة (Parallel Processing)
- إمكانية الاستخدام على الموبايل
- دعم خوارزميات جديدة