المحاضرة الخامسة: مراحل البرمجة في الذكاء الاصطناعي

تُركز هذه المحاضرة على تفكيك العملية المنهجية لتطوير ونشر نماذج الذكاء الاصطناعي، وهي العملية التي تجمع بين مبادئ البرمجة التقليدية (الخوارزميات، التسلسل، الدقة) وبين خوارزميات التعلم الآلي المعقدة، ويُشار إلى هذه العملية غالبًا باسم دورة حياة التعلم الآلي(Machine Learning Lifecycle) ، أو MLOps (Machine Learning Operations)، وهي تتطلب مجموعة متكاملة من المهارات الهندسية والتحليلية التي يجب على الطالب إتقانها.

1. تحديد الهدف وتحليل المتطلبات(Problem Definition)

تبدأ عملية برمجة أي نظام ذكي بتحديد دقيق وواضح للمشكلة المراد حلها، ويجب على المبرمج والطالب تحويل التحدي العام (مثل "تحليل الرأي العام") إلى هدف كمّي قابل للقياس، مثل "بناء نموذج قادر على تصنيف المشاعر بنسبة دقة 90%"، وهذه المرحلة هي مرحلة التفكير النقدي التي يتم فيها تعريف:

- **نوع المشكلة:** هل هي مشكلة تصنيف(Classification) ، أم انحدار(Regression) ، أم تجميع(Clustering) ، أم معالجة لغة طبيعية(NLP) ، ويتوقف اختيار الخوارزمية والأدوات المناسبة (المذكورة في المحاضرة 4) على هذا التحديد الدقيق.

إن تحديد نوع المشكلة هو الخطوة التأسيسية التي تُملي المسار التكنولوجي والمنهجي بالكامل لمشروع الذكاء الاصطناعي، إذ يتوجب على المبرمجين والمهندسين تحويل التحدي الواقعي إلى إطار رياضي أو إحصائي قابل للحل، فإذا كان الهدف هو فرز رسائل البريد الإلكتروني الواردة إلى "مُزعجة" أو "غير مُزعجة"، فنحن أمام مشكلة تصنيف (Classification)، حيث يقوم النموذج بتعيين مدخل (الرسالة) إلى إحدى الفئات المحددة مسبقاً، أما إذا كان الهدف هو التنبؤ بسعر منزل بناءً على مساحته وعدد غرفه، فهنا تظهر مشكلة الانحدار(Regression) ، وهي تتطلب التنبؤ بقيمة مستمرة وليست فئة منفصلة، وفي المقابل، إذا كان المطلوب هو تقسيم العملاء إلى مجموعات بناءً على أنماط شرائية متشابهة دون وجود تسميات مسبقة للفئات، نكون أمام مشكلة تجميع(Clustering) ، والتي تندرج تحت التعلم غير المراقب، ويتعدى تحديد النوع هذا الجانب ليطال المجالات التخصصية مثل معالجة اللغة الطبيعية (NLP) للتعامل مع النصوص، أو الرؤية الحاسوبية (Computer Vision) للصور، ويُعد هذا التحديد المُنظم للنوع هو المفتاح لاختيار الخوارزمية المناسبة (كالشبكات العصبونية الالتفافية لـ CV أو الشبكات المتكررة لـ NLP) واختيار أطر العمل البرمجية اللازمة، مما يضمن أن تكون جهود البرمجة مركزة ومحققة لخاصية الدقة.

نوع المشكلة	الهدف الرئيسي	مثال تطبيقي
التصنيف(Classification)	التنبؤ بفئة منفصلة ومحدودة	تحديد ما إذا كانت الصورة تحتوي على كلب أم قط،
الانحدار(Regression)	التنبؤ بقيمة رقمية مستمرة	التنبؤ بدرجة حرارة الغد،
التجميع(Clustering)	تجميع نقاط البيانات المتشابهة	تقسيم الجمهور المستهدف حسب السلوك،
معالجة اللغة الطبيعية(NLP)	فهم وإنشاء ومعالجة لغة البشر	ترجمة النصوص أو تحليل المشاعر،

- **مقاييس الأداء(Metrics)** : تحديد المقاييس التي سيتم استخدامها لتقييم صحة النموذج (مثل الدقة، الاستدعاء، أو الخطأ المتوسط)، وهذا يضمن أن تتوافق نتائج النموذج مع متطلبات **الصحة (Correctness)** البرمجية.

إن تحديد مقاييس الأداء (Metrics) يمثل جسر الربط بين الأداء الرياضي للنموذج والنجاح العملي في حل المشكلة المحددة، وهو يضمن أن تتوافق نتائج النموذج مع متطلبات الصحة (Correctness) البرمجية والوظيفية، فليست كل المقاييس متساوية الأهمية، ويعتمد الاختيار الحكيم للمقياس على سياق المشكلة ونوعها، ففي مشكلة التصنيف المتوازنة، قد تكون الدقة (Accuracy) وهي النسبة المئوية للتنبؤات الصحيحة، كافية لتقييم الصحة الإجمالية، ولكن

في الأنظمة التي تتطلب حساسية عالية (مثل أنظمة الكشف عن الأمراض النادرة أو الاحتيال)، يُصبح مقياس الاستدعاء (Recall) أو الحساسية أكثر أهمية، وهو يقيس قدرة النموذج على العثور على جميع الحالات الإيجابية الحقيقية، بينما يقيس مقياس الدقة الموجهة (Precision) كمية التنبؤات الإيجابية التي كانت صحيحة بالفعل، مما يقلل من الإنذارات الكاذبة، وفي المقابل، في مشكلات الانحدار، يتم استخدام مقاييس تركز على قياس حجم الخطأ بين القيمة المتوقعة والقيمة الفعلية، مثل الخطأ المتوسط المطلق (MAE) أو جذر متوسط مربع الخطأ(RMSE) ، ويجب تحديد هذه المقاييس في مرحلة التصميم لتوجيه عملية تدريب النموذج وتوفير معيار كمّي يتم من خلاله الحكم على مدى نجاح خوارزميات التدريب والتحسين في مرحلة لاحقة.

المقياس	نوع المشكلة	الوصف والمغزى
الدقة(Accuracy)	التصنيف	النسبة المئوية للتنبؤات الصحيحة بشكل عام (النماذج المتوازنة).
الاستدعاء(Recall)	التصنيف	قدرة النموذج على تحديد جميع الحالات الإيجابية (هام للكشف عن حالات نادرة).
الخطأ المتوسط المطلق(MAE)	الانحدار	متوسط الفرق المطلق بين القيمة المتوقعة والفعلية (سهل التفسير).
جذر متوسط مربع الخطأ(RMSE)	الانحدار	يعطي وزناً أكبر للأخطاء الكبيرة (مفيد في تجنب الانحرافات الشديدة).

- **البيانات المطلوبة:** تحديد طبيعة البيانات المصدرية المطلوبة والكمية اللازمة لتدريب النموذج، وهي التي ستشكل أساس مرحلة هندسة البيانات اللاحقة.

تُعد مرحلة تحديد البيانات المطلوبة بمثابة رسم خريطة الطريق لمرحلة هندسة البيانات والمعالجة المسبقة، إذ أن جودة البيانات وكميتها هما اللبنة الأساسية التي تُبنى عليها "خبرة" النموذج الذكي، ويتطلب هذا التحديد تحليل شامل لطبيعة البيانات المصدرية، بمعنى هل هي بيانات نصية غير منظمة، صور، سجلات قاعدة بيانات رقمية، أم خليط من أنواع متعددة، كما يجب تقدير الكمية اللازمة لتدريب النموذج بنجاح، ففي مهام التعلم العميق المعقدة مثل الرؤية الحاسوبية، قد يتطلب الأمر آلاف أو ملايين الأمثلة لضمان قدرة النموذج على تحقيق خاصية العمومية (Generality)، ومن هنا، يجب تحديد مصادر هذه البيانات (قواعد بيانات داخلية، مستودعات عامة، أو عمليات جمع جديدة)، وضمان أنها بيانات مُعلّمة بشكل صحيح إذا كانت المشكلة تصنيف أو انحدار (تعلم مُراقب)، ويجب أن يأخذ هذا التحليل في الاعتبار القضايا المتعلقة بجودة البيانات مثل الاكتمال والتناسق والتمثيل، لأن أي تحيز أو ضوضاء في البيانات المطلوبة سينعكس سلباً على أداء النموذج، مما يُبرز أهمية هذه المرحلة في تشكيل أساس متين لعمليات البرمجة اللاحقة المتعلقة بالتنظيف والتطبيع واستخراج الميزات.

خاصية البيانات	الأهمية في البرمجة	مثال على المتطلبات
الطبيعة(Nature)	تؤثر على مكتبات البرمجة (مثل NumPy للأرقام، Pillowللصور)	بيانات صور لتدريب نموذج رؤية حاسوبية.
الكمية(Volume)	تحدد قدرة النموذج على التعميم وتجنب فرط التخصيص	ملايين الأمثلة في مشروع تحليل المشاعر اللغوية.
الجودة(Quality)	تؤثر مباشرة على دقة النموذج وصحته	خلو البيانات من القيم المفقودة والأخطاء المتكررة.
التصنيف(Labeling)	ضرورية للتعلم المراقب (تصنيف/انحدار)	كل صورة مصنفة بدقة باسم الكائن الموجود بداخلها.

2. هندسة البيانات والمعالجة المسبقة(Data Engineering & Preprocessing)

تُعد هذه المرحلة هي الأكثر اعتماداً على مهارات **البرمجة التقليدية**، حيث تتطلب كتابة شفرات برمجية للتعامل مع البيانات الضخمة التي ستُستخدم ك "خبرة" للنموذج الذكي، وتنقسم هذه المرحلة إلى خطوات متسلسلة لضمان جودة المدخلات:

أ. الجمع والتكامل(Collection and Integration) :

استخدام لغات البرمجة (مثل Python ومكتباتها (مثل Pandas لجمع البيانات من مصادر مختلفة ودمجها في مجموعة بيانات موحدة، وهذه الخطوة يجب أن تلتزم بـ خاصية التسلسل(Sequence) ، فلا يمكن معالجة البيانات قبل جمعها.

تُعد مرحلة الجمع والتكامل هي نقطة الانطلاق الحاسمة لدورة حياة التعلم الآلي، إذ أنها تتطلب تطبيقاً صارماً لخاصية التسلسل (Sequence) البرمجية، حيث لا يمكن البدء بأي عمليات معالجة أو تدريب قبل التأكد من جمع ودمج كافة مصادر البيانات المطلوبة بشكل كامل ومنظم، ويتضمن هذا العمل استخدام لغات برمجة قوية مثل Python، والاستعانة بمكتبات متخصصة كـ Pandاس للتعامل مع هياكل البيانات الجداولية، أو Beautiful Soup لسحب البيانات من الويب(Web Scraping) ، أو حتى واجهات برمجة تطبيقات مخصصة (APIs) لاستخلاص البيانات من خدمات الشركات، وتتطلب هذه المرحلة مهارات هندسية عالية لضمان التوصيل الآمن والموثوق بقواعد البيانات العلائقية (SQL) أو غير العلائقية (NoSQL) أو مستودعات التخزين السحابية، والهدف النهائي هو توحيد هذه التدفقات المتنوعة من البيانات، والتي قد تأتي بتنسيقات مختلفةCSV) ، JSON، (XML، في مجموعة بيانات موحدة ومتسقة (Unified Dataset)، مما يزيل التناقضات الأولية في التنسيق ويجهز البيانات للمراحل التالية، وهذه العملية تُشبه بناء الأساس الذي سيُبنى عليه النموذج الذكي، وأي قصور في الجمع أو التكامل سيؤدي حتماً إلى نموذج ضعيف وغير موثوق.

مصدر البيانات	أداة البرمجة النموذجية	التنسيق المشترك
قواعد البيانات(SQL/NoSQL)	موصلات Python مثل psycopg2 أو(pymongo	جداول منظمة، BSON
واجهات الAPIs	مكتبة Requests في Python	XMLأوJSON
الملفات المنظمة	مكتبة Pandas لقراءة ملفاتCSV/Excel	XLSX ،CSV
الويب(Unstructured)	مكتبة Beautiful Soup للWeb Scraping	HTML

ب. التنظيف والمعالجة(Cleaning and Transformation) :

تُعد مرحلة **التنظيف والمعالجة** هي المرحلة الأكثر أهمية لضمان جودة المدخلات، وهي تتطلب تطبيق قواعد برمجية صارمة **للتخلص من الأخطاء والضوضاء** التي قد تضلل خوارزميات التعلم الآلي، ويتمحور جزء كبير من هذا العمل حول التعامل مع مشكلة **القيم المفقودة(Missing Values)** ، حيث يجب على المبرمج كتابة خوارزميات تحدد طريقة التعويض عنها (مثل استخدام المتوسط **Mean**أو الوسيط **Median**في عملية تُسمى **Imputation**، أو اتخاذ قرار بحذف السجلات الناقصة بالكامل، ويجب أن يكون هذا القرار قائماً على تحليل كمية ونمط القيم المفقودة، وخطوة أخرى حاسمة هي **التطبيع (Normalization)** أو **القياس(Scaling)** ، وهي عملية برمجية لتحويل قيم الميزات إلى نطاق موحد (غالباً بين 0 و 1 أو بمتوسط صفر وانحراف معياري واحد)، وهذا يضمن أن جميع الميزات تساهم بالتساوي في عملية التعلم، خاصة في الخوارزميات الحساسة للمقياس مثل الشبكات العصبونية أو آلة المتجهات الداعمة، مما يضمن **الفاعلية** في التدريب ويمنع أن تطغى ميزة ذات نطاق قيم كبير على ميزة أخرى ذات نطاق قيم صغير، وتشمل المعالجة أيضاً التعامل مع القيم المتطرفة **(Outliers)**وتنويع ترميز البيانات الفئوية **(Encoding)** باستخدام دوال برمجية.

ج. هندسة الميزات(Feature Engineering) :

وهي عملية إبداعية ومهارية يقوم فيها المبرمج بإنشاء ميزات (Features) جديدة وذات مغزى من البيانات الخام الموجودة لتحسين أداء النموذج، فمثلاً، في تحليل النصوص، يتم استخراج ميزة **طول الجملة** أو **تكرار الكلمة** لتقديمها كمدخلات إضافية للنموذج، وهذا يمثل تطبيقاً متقدماً لـ **خاصية الإخراج (Output)** البرمجية.

تُمثل هندسة الميزات (Feature Engineering) الجانب الإبداعي والمهاري لعملية إعداد البيانات، إذ لا يقتصر دور المبرمج هنا على تنظيف البيانات، بل يتعداه إلى إنشاء ميزات (Features) جديدة وذات مغزى لم تكن موجودة بشكل صريح في البيانات الخام، وهذا هو التطبيق المتقدم لخاصية الإخراج (Output) البرمجية، حيث يتم استخدام المنطق والخوارزميات لمعالجة المدخلات الحالية وتوليد مخرجات جديدة ذات قيمة تنبؤية أعلى للنموذج، فمثلاً، بدلاً من إدخال تاريخ الشراء كسجل خام، يمكن للمبرمج إنشاء ميزات جديدة مثل "يوم الأسبوع"، أو "المدة منذ آخر شراء"، أو "معدل الشراء خلال آخر 30 يوماً"، وفي سياق تحليل النصوص(NLP) ، يتضمن هذا استخراج ميزات مثل "طول الجملة"، أو "تردد الكلمات غير المتوقعة"(TF-IDF) ، أو إنشاء متغيرات ثنائية (Dummy Variables)لتمثيل الفئات، ويتطلب هذا العمل فهماً عميقاً لمجال المشكلة (Domain Knowledge)لتحديد العلاقات الخفية بين المتغيرات، وكلما كانت الميزات المُهندسة أفضل، كلما قل الاعتماد على خوارزميات التعلم الآلي المعقدة، وارتفعت قدرة النموذج على التنبؤ بدقة، مما يُحسن أداء النموذج بشكل كبير دون الحاجة إلى تغيير هيكله،

عملية هندسة الميزات	الوصف البرمجي	المثال التحليلي
التجميع(Aggregation)	استخدام دوال التجميع (مثل المتوسط أو المجموع) على نوافذ زمنية أو فئات	حساب متوسط إنفاق العميل في الشهر الماضي،
التحويل (Transformation)	استخدام دوال رياضية (مثل اللوغاريتم أو التربيع) لتغيير توزيع الميزة	تحويل ميزة ذات توزيع منحرف لتقريب التوزيع الطبيعي،
التقسيم(Binning)	تقسيم نطاق مستمر إلى فئات منفصلة	تقسيم الأعمار إلى فئات (مثل 18-25، 26-40)،
التفاعل(Interaction)	ضرب أو جمع ميزتين معاً لإنشاء ميزة جديدة	ضرب الطول في العرض للحصول على ميزة المساحة،

3. بناء النموذج والتدريب(Modeling and Training)

في هذه المرحلة، يتم استخدام أطر عمل التعلم العميق TensorFlow) أو PyTorch) التي ذكرناها في المحاضرة 4 لتحويل البيانات المُعدة إلى نموذج ذكي قادر على التنبؤ، وتتطلب هذه المرحلة تطبيقاً دقيقاً للمنطق الرياضي:

أ. تقسيم البيانات(Data Splitting) :

يتم تقسيم مجموعة البيانات النهائية برمجياً إلى ثلاث مجموعات:

- **التدريب(Training Set) :** الجزء الأكبر الذي يتعلم منه النموذج الأنماط.
- **التحقق(Validation Set) :** يستخدم لضبط معاملات النموذج (Hyperparameters) أثناء التدريب.
- **الاختبار(Test Set)** : يستخدم لتقييم الأداء النهائي للنموذج بشكل محايد، مما يضمن **الصحة (Correctness)**.

تُمثل عملية **تقسيم البيانات** إجراءً برمجياً منهجياً بالغ الأهمية لضمان **الصحة (Correctness)** البرمجية والتعميم الفعال للنموذج، إذ يتم تقسيم مجموعة البيانات المُعَدة والمُنظفة إلى ثلاثة أجزاء متميزة وظيفياً لخدمة أغراض مختلفة في دورة التدريب، الجزء الأول وهو **مجموعة التدريب(Training Set)** ، يمثل الحصة الأكبر من البيانات (غالباً 60% إلى 80%)، وهو الجزء الذي يتعلم منه النموذج الأنماط والعلاقات عبر خوارزميات التعلم الآلي، ويتم تعديل أوزان النموذج الداخلية **(Weights)**استناداً إلى الأخطاء المُحتسبة على هذه المجموعة، والجزء الثاني هو **مجموعة التحقق(Validation Set)** ، ويُستخدم هذا الجزء لضبط المعاملات الفائقة للنموذج

(Hyperparameters)، مثل عدد الطبقات أو حجم الدفعة **(Batch Size)**، وتُعد مجموعة التحقق بمثابة عين المبرمج أثناء عملية التدريب، حيث تساعده في اتخاذ القرارات المتعلقة بإيقاف التدريب المبكر **(Early Stopping)**لتجنب **الإفراط في التخصيص (Overfitting)** للبيانات، أما الجزء الثالث والأكثر أهمية لتقييم الصحة، فهو **مجموعة الاختبار(Test Set)** ، وهي مجموعة تُحفظ بشكل سري ولا تُستخدم إطلاقاً أثناء مرحلتي التدريب أو التحقق، ويتم استخدامها فقط لتقييم الأداء النهائي للنموذج بشكل **محايد** وواقعي قبل نشره، مما يوفر مقياساً حقيقياً لقدرة النموذج على العمل على بيانات جديدة لم يرها من قبل.

مجموعة البيانات	الغرض الوظيفي	النسبة التقريبية
التدريب(Training Set)	تعلم الأنماط وتحديث الأوزان الداخلية للنموذج	60% - 80%
التحقق(Validation Set)	ضبط المعاملات الفائقة واكتشاف الإفراط في التخصيص	10% - 20%
الاختبار(Test Set)	تقييم الأداء النهائي والعمومية بشكل محايد	10% - 20%

ب. تصميم الهيكل(Architecture Design) :

كتابة الشفرة اللازمة لتحديد هيكل الشبكة العصبونية، بما في ذلك عدد الطبقات، ونوع الطبقات (مثل طبقات الالتفاف Conv2D في الرؤية الحاسوبية)، وتحديد دالة الخسارة (Loss Function) والمحسِّن(Optimizer) ، وتعتمد جودة هذا التصميم على مهارة المبرمج في تطبيق **الدقة والمنطق**.

يتطلب تصميم الهيكل كتابة شفرة برمجية معقدة وباستخدام أطر عمل متخصصة (مثل TensorFlow أو PyTorch)لإنشاء البنية الأساسية للشبكة العصبونية، وتتجسد هنا مهارة المبرمج في تطبيق الدقة والمنطق الرياضي والهندسي، ففي مشاريع الرؤية الحاسوبية(Computer Vision) ، يتضمن التصميم تحديد طبقات الالتفاف Conv2D) (Layersوعدد المرشحات (Filters)وحجم خطوة الالتفاف (Stride)، أما في مشاريع معالجة اللغة الطبيعية(NLP) ، فيتم التركيز على طبقات الشبكات المتكررة (Recurrent Layers)أو آليات الانتباه (Attention Mechanisms)، ولا يقتصر التصميم على الطبقات فحسب، بل يشمل أيضاً تحديد دالة الهدف أو دالة الخسارة(Loss Function) ، وهي الدالة التي تُقيس مدى ابتعاد تنبؤات النموذج عن القيم الفعلية (مثل Binary Cross-Entropyللتصنيف الثنائي أو Mean Squared Errorللانحدار)، وتُستخدم هذه القيمة لتوجيه عملية التعلم، وبالإضافة إلى ذلك، يجب تحديد المُحسِّن(Optimizer) ، وهو الخوارزمية (مثل Adamأو SGD)المسؤولة عن تحديث أوزان النموذج لتقليل دالة الخسارة، وتعتمد جودة هذا الهيكل على المنطق الرياضي المستخدم لضمان أن يكون النموذج قادراً على التعلم بكفاءة وفي الوقت المناسب دون الوقوع في الحد الأدنى المحلي (Local Minima)، مما يجعل هذه المرحلة دقيقة وحاسمة لنجاح المشروع.

عنصر الهيكل	الدور البرمجي/الوظيفي	مثال برمجي شائع
نوع الطبقة	معالجة المدخلات واستخراج الميزات (الأنماط)	طبقة الالتفاف Conv2D للصور، طبقة LSTM للنصوص،
دالة الخسارة (Loss Function)	قياس خطأ النموذج لتوجيه التعلم	Categorical Cross-Entropyللتصنيف متعدد الفئات،
المُحسِّن(Optimizer)	تحديد كيفية تعديل الأوزان لتقليل الخسارة	خوارزمية Adam لتسريع عملية التقارب،
دالة التنشيط (Activation Function)	إدخال اللاخطية في النموذج لتمكين تعلم العلاقات المعقدة	دالة ReLU أوSigmoid ،

ج. مرحلة التكرار والتحسين(Iteration and Refinement) :

لا يتم تدريب النموذج مرة واحدة، بل تتضمن هذه المرحلة دورات **تكرارية (Loops)** من التدريب والتحسين، حيث يراقب المبرمج أداء النموذج ويقوم بتعديل المعاملات (مثل معدل التعلم Learning Rate) وإعادة التدريب حتى يصل النموذج إلى مستوى الأداء المطلوب.

تؤكد مرحلة التكرار والتحسين على أن تدريب نماذج الذكاء الاصطناعي هي عملية تكرارية (Iterative) وليست خطوة واحدة نهائية، وهي تتضمن دورات (Loops) متكررة يقوم فيها المبرمج بمراقبة أداء النموذج على مجموعة التدريب والتحقق بشكل مستمر، ويتم ذلك عبر مراقبة قيمة دالة الخسارة والمقاييس المحددة لكل عهد (Epoch) من التدريب، ويقوم المبرمج باتخاذ القرارات المتعلقة بتعديل المعاملات الفائقة بناءً على ملاحظاته، فإذا كان الأداء ضعيفاً على مجموعة التدريب، قد يحتاج المبرمج إلى زيادة تعقيد النموذج أو زيادة معدل التعلم(Learning Rate) ، وإذا كان النموذج يُظهر إفراطاً في التخصيص (Overfitting) أداء ممتاز على التدريب وضعيف على التحقق)، فيجب عليه تطبيق تقنيات تنظيمية مثل التسرب (Dropout) أو تقليل معدل التعلم، وتشمل عملية التحسين أيضاً تجربة هياكل مختلفة أو دمج ميزات جديدة من مرحلة هندسة الميزات، ولا يُعد النموذج جاهزاً للانتقال إلى مرحلة التقييم إلا بعد أن يصل إلى مستوى الأداء المطلوب على مجموعة التحقق، مما يضمن أن النموذج يتمتع بأقصى درجات الدقة والفاعلية الممكنة قبل استخدامه في بيئة الإنتاج.

سيناريو الأداء	الإجراء البرمجي المقترح	الهدف من الإجراء
إفراط في التخصيص(Overfitting)	تطبيق التسرب (Dropout) أو تقليل معدل التعلم	تحسين قدرة النموذج على التعميم،
نقص في التخصيص(Underfitting)	زيادة عمق الشبكة أو زيادة زمن التدريب	زيادة قدرة النموذج على تعلم الأنماط المعقدة،
التقارب البطيء (Slow Convergence)	زيادة معدل التعلم (Learning Rate) أو تغيير المُحسّن	تسريع الوصول إلى الحد الأدنى للخسارة،
تذبذب عالي(High Variance)	استخدام أحجام دفعات (Batch Sizes) أكبر	استقرار عملية تحديث الأوزان،

4. التقييم والتحقق(Evaluation)

بعد الانتهاء من التدريب، يتم استخدام مجموعة الاختبار (Test Set) لتقييم قدرة النموذج على العمل على بيانات جديدة لم يرها من قبل، وهذا هو الاختبار الحقيقي لقدرة النموذج على تحقيق خاصية **العمومية(Generality)** ، وتشمل هذه المرحلة ما يلي:

أ. حساب المقاييس(Metric Calculation) :

تُعد مرحلة **حساب المقاييس** تطبيقاً برمجياً دقيقاً لتقييم مدى تحقيق النموذج لخاصية **العمومية (Generality)** على بيانات الاختبار التي لم يرها من قبل، وتتطلب هذه المرحلة كتابة دوال برمجية متخصصة، غالبًا ما تستخدم مكتبات مثل **Scikit-learn**في Python ، لبناء وتحليل **المصفوفة المختلطة(Confusion Matrix)** ، وهي في جوهرها جدول يلخص أداء نموذج التصنيف من خلال تحديد أربعة احتمالات أساسية لكل فئة :التنبؤات الإيجابية الحقيقية **(True Positives)**، والسلبية الحقيقية **(True Negatives)**، والإيجابية الخاطئة **(False Positives)**، والسلبية الخاطئة **(False Negatives)**، ومن هذه المصفوفة، يتم اشتقاق المقاييس الرئيسية، حيث تُحسب **الدقة (Accuracy)**كنسبة التنبؤات الصحيحة بشكل عام إلى إجمالي التنبؤات، وهي مقياس جيد لتقييم الصحة الإجمالية في مجموعات البيانات المتوازنة، ولكن في الأنظمة التي تكون فيها الفئات غير متوازنة (مثل الكشف عن الاحتيال الذي يمثل نسبة ضئيلة)، نتحول إلى مقاييس أكثر دقة مثل **القياس الموجه(Precision)** ، الذي يقيس مدى صحة التنبؤات

الإيجابية للنموذج (تقليل الإنذارات الكاذبة)، والاستدعاء(Recall) ، الذي يقيس قدرة النموذج على العثور على جميع الحالات الإيجابية الفعلية، مما يضمن أن هذه المقاييس الموجهة تعكس متطلبات العمل الفعلية للنموذج.

المقياس	الوصف الرياضي (من المصفوفة المختلطة)	الاستخدام الأمثل
الإيجابية الحقيقية(TP)	الحالات التي تنبأ بها النموذج بشكل صحيح كإيجابية	الأساس لحساب الاستدعاء والقياس
الدقة(Accuracy)	(TP + TN) / Total	تقييم الأداء العام في البيانات المتوازنة
القياس الموجه (Precision)	TP / (TP + FP)	الأنظمة التي تتطلب تقليل الإنذارات الكاذبة (مثلاً: تشخيص طبي دقيق)
الاستدعاء(Recall)	TP / (TP + FN)	الأنظمة التي تتطلب عدم تفويت الحالات الإيجابية (مثلاً: الكشف عن احتيال)

ب. التفسير والتوثيق(Interpretation and Documentation) :

تفسير النتائج المحسوبة والإجابة على سؤال: هل النموذج جيد بما يكفي لـ حل المشكلة التي تم تعريفها في البداية؟ ويجب توثيق جميع الخطوات والنتائج بشكل منهجي ومنطقي.

تُمثل مرحلة التفسير والتوثيق تتويجاً منطقياً وعملياً لجهود البرمجة والتدريب، إذ لا يكفي مجرد حساب المقاييس، بل يجب تفسير هذه النتائج المحسوبة في سياق المشكلة الأصلية التي تم تعريفها في المرحلة الأولى، وتتم الإجابة بشكل نقدي ومنطقي على السؤال الحاسم :هل النموذج جيد بما يكفي لحل المشكلة؟، فإذا كان الهدف هو تحقيق دقة 90% وتم تحقيق 88%، يجب توثيق هذا الفارق وتفسيره، وتتطلب هذه المرحلة تحويل الأرقام المجردة إلى تقييمات وظيفية، مع تحديد نقاط القوة في الأداء (مثل الأداء الممتاز في فئة معينة) ونقاط الضعف التي تحتاج إلى تحسين في دورات تدريبية مستقبلية، أما عملية التوثيق(Documentation) ، فهي لا تقل أهمية، حيث يجب توثيق جميع خطوات دورة الحياة بشكل منهجي، بدءاً من مصادر البيانات المستخدمة، مروراً بالمعاملات الفائقة النهائية المستخدمة (Final Hyperparameters)، وصولاً إلى نتائج الأداء الكاملة على مجموعة الاختبار، ويضمن هذا التوثيق الشفافية (Transparency)والدقة البرمجية، حيث يسمح للمطورين الآخرين أو المراجعين بإعادة إنتاج النتائج (Reproducibility)، ويشكل أساساً لخطط الصيانة والنشر اللاحقة.

عنصر التوثيق	محتواه البرمجي/التحليلي	الغرض منه
تقرير الأداء النهائي	جدول يضم Accuracy, Precision, Recall على Test Set	التحقق من صحة (Correctness) النموذج قبل النشر
المعاملات الفائقة (Hyperparameters)	قائمة بالإعدادات النهائية للنموذج (Learning Rate, Epochs, Batch Size)	ضمان قابلية إعادة الإنتاج (Reproducibility)
تحليل الأخطاء(Error Analysis)	أمثلة على التنبؤات الخاطئة الأكثر شيوعاً (FP وFN)	تحديد نقاط الضعف لتوجيه إعادة التدريب المستقبلي
خطة النشر المقترحة	متطلبات الموارد اللازمة (CPU/GPU) للتشغيل في بيئة الإنتاج	التخطيط لمرحلة النشر والصيانة (Deployment)

5. النشر والصيانة(Deployment and Maintenance)

وهي المرحلة النهائية التي ينتقل فيها النموذج المدرب إلى مرحلة الاستخدام العملي ضمن **التطبيقات الرقمية** أو **الروبوتات**، وهنا تتجلى أهمية البرمجة في بيئة الإنتاج:

أ. التغليف والواجهة البرمجية(Packaging and API) :

استخدام لغات البرمجة (مثل Python مع إطارات عمل الويب مثل Flask أوDjango) لتغليف النموذج في خدمة قابلة للاستدعاء عبر واجهة برمجية(REST API) ، وهذا يضمن أن يتمكن أي تطبيق خارجي من استخدامه، مع الالتزام بـ **فاعلية** وسرعة الاستجابة.

ب. المراقبة وإدارة الإصدارات(Monitoring and Version Control) :

كتابة شفرات برمجية لمراقبة أداء النموذج في بيئة الإنتاج بشكل مستمر، فربما تتغير الأنماط الجماهيرية بمرور الوقت، مما يؤدي إلى تدهور أداء النموذج(Model Drift) ، وتُستخدم أنظمة إدارة الإصدارات (مثلGit) لضمان أن تكون جميع مراحل الكود قابلة للتتبع والعودة إلى إصدارات سابقة، وهذا ضروري لضمان **الشفافية والدقة** في عمل النظام الذكي.

ج. إعادة التدريب والأتمتة(Retraining and Automation) :

برمجة دورة آلية لإعادة تدريب النموذج وتحديثه باستخدام البيانات الجديدة بشكل دوري، مما يضمن أن النموذج يظل **قادراً على التعلم والتكيف** مع التغيرات البيئية الجديدة.

تمارين وتطبيقات عملية (دورة حياة النموذج)

التمرين 1: تحديد المرحلة والأداة(NLP Example)

أمامك الخطوات التالية لتطوير نموذج لمعالجة اللغة الطبيعية:

- الخطوة أ: استخدام مكتبة Pandas لحذف جميع علامات الترقيم من التعليقات النصية.
- الخطوة ب: تحديد أن النموذج يعمل بشكل أفضل مع شبكة عصبونية ذات 5 طبقات (Layers5).
- الخطوة ج: حساب نسبة الدقة التي حققها النموذج على بيانات الاختبار.
- الخطوة د: استخدام إطار عمل Flask لإنشاء نقطة نهاية (Endpoint) للنموذج.

السؤال :حدد مرحلة دورة حياة التعلم الآلي التي تنتمي إليها كل خطوة:

- الخطوة أ :هندسة البيانات والمعالجة المسبقة.
- الخطوة ب: بناء النموذج والتدريب (تصميم الهيكل).
- الخطوة ج :التقييم والتحقق (حساب المقاييس).
- الخطوة د: النشر والصيانة (التغليف والواجهة البرمجية).

التمرين 2: تحدي الفاعلية في مرحلة النشر

إذا كان فريقك قد قام بتدريب نموذج ذكاء اصطناعي لديه دقة عالية (95%) ولكنه بطيء جداً، حيث يستغرق 3 ثوانٍ لمعالجة كل طلب، وتم نشره كنظام لتصنيف الرسائل العاجلة للرئيس التنفيذي، ما هو الخطر المحتمل، وكيف يمكن للبرمجة الموجهة لـ C++ أن تساعد؟

1. **الخطر:** الخطر هو إهمال خاصية **الفاعلية(Efficiency)** ، حيث أن زمن الاستجابة البطيء يعني أن الرسائل العاجلة لن تصل في الوقت الفعلي، مما يؤدي إلى فشل النظام في مهمة **إدارة الأزمات والتواصل الاستراتيجي**.
2. **دور++C :** يمكن للمبرمجين إعادة كتابة الأجزاء الأكثر استهلاكاً للوقت في النموذج باستخدام) C++ لتكون نواة سريعة) والوصول إليها عبر Python ، مما يقلل زمن الاستجابة إلى أجزاء من الثانية دون التضحية بالدقة، وهو دمج حيوي بين لغات البرمجة المختلفة.