الوحدة الثانية: العلاقة والتنفيذ بلغات البرمجة

المحاضرة الثالثة: الذكاء الاصطناعي ودور أنظمة البرمجة

تتناول هذه المحاضرة الجوانب الوظيفية والمنهجية التي تربط بين البرمجة، بمفاهيمها الأساسية (التسلسل، المنطق، الفاعلية)، وبين أنظمة الذكاء الاصطناعي، لتوضيح كيف تدعم الأولى الثانية في مراحل التطوير والتنفيذ.

1. وظائف البرمجة في دعم أنظمة الذكاء الاصطناعي(AI Support Functions) :

الذكاء الاصطناعي، على الرغم من اعتماده على خوارزميات التعلم المعقدة، لا يمكن أن يعمل بمعزل عن البرمجة التقليدية، فالبرمجة هي البنية التحتية والمنهجية التي تمكّن خوارزميات التعلم الآلي والعميق من الوجود والعمل بفاعلية، ولا يقتصر دور المبرمج على "تدريب" النموذج وحسب، بل يشمل هندسة العملية برمتها من البداية إلى النهاية.

أ. هندسة البيانات(Data Engineering) :

يُعد دور هندسة البيانات هو الأساس الذي يقوم عليه صرح الذكاء الاصطناعي بأكمله، وإذا كانت البيانات هي "وقود" نماذج التعلم الآلي، فإن مهندس البيانات هو من يقوم ببناء وصيانة خطوط الأنابيب (Pipelines) التي تنقل هذا الوقود وتنقية، ولضمان وصوله نظيفًا وفعالًا إلى "محرك" الخوارزميات، ولا يمكن لنموذج أن يحقق خاصية الدقة أو العمومية المرجوة دون بنية تحتية برمجية متينة وموثوقة للبيانات، وتتجسد البرمجة في هذا المجال في بناء ما يُعرف بـ أنابيب استخلاص-تحويل-تحميل (ETL) أو استخلاص-تحميل-تحويل(ELT) ، وهي عبارة عن مجموعة معقدة من الشفرات المصممة لأداء وظائف مستمرة ومتكررة وذلك لتنفيذ المهام التالية:

أولا: التجميع والاستخلاص (Extraction) البرمجي: فتح صنبور البيانات

تتطلب عملية استخلاص البيانات (Data Extraction) كتابة شفرات برمجية دقيقة للتفاعل مع مصادر بيانات متنوعة وغير متجانسة، وتُستخدم Python كأداة أساسية هنا نظرًا لمرونتها وغنى مكتباتها، والهدف هو "سحب" البيانات الأولية من مكان وجودها وتحويلها إلى هيكل يمكن معالجته.

- **الاتصال بقواعد البيانات :(Databases)** يتضمن ذلك كتابة شفرات تستخدم مكتبات مثل psycopg2 للاتصال بقواعد بيانات **SQL**العلائقية (مثلPostgreSQL) لتنفيذ استعلامات SELECTمعقدة أو استخدام مكتبات مثل PyMongo للتفاعل مع قواعد بيانات **NoSQL**غير العلائقية (مثلMongoDB)، ويجب أن يتضمن الكود آليات لمعالجة الأخطاء (Error Handling) وفصل الاتصال بعد الانتهاء لضمان كفاءة النظام،

مثال: إذا كنا نبني نظام توصية لمتجر إلكتروني، فإن شفرة الاستخلاص يجب أن تتصل بجدول **SQL**لسحب سجلات تاريخ الشراء وتصنيفات المنتجات، وربما بقاعدة بيانات **NoSQL** لسحب بيانات تفاعل المستخدمين الفورية (نقرات، وقت البقاء على الصفحة).

- **استخدام واجهات برمجة التطبيقات(APIs) :** غالبًا ما تكون البيانات الخارجية، مثل أسعار الأسهم، أو بيانات الطقس، أو التفاعلات على وسائل التواصل الاجتماعي، متاحة فقط عبر واجهات برمجية للتطبيق(APIs) ، وتتطلب البرمجة استخدام مكتبات مثل requests في Python لإنشاء طلبات **HTTP (GET/POST)**، والتعامل مع رموز المصادقة(Authentication Tokens) ، ومعالجة الاستجابات الواردة غالبًا بصيغة **JSON**، ويجب على الشفرة أن تحترم قيود المعدل (Rate Limits) التي تفرضها واجهة API لتجنب حظر النظام،

مثال: لاستخلاص تغريدات متعلقة بمنتج معين لتحليل المشاعر(Sentiment Analysis) ، يجب كتابة كود يتفاعل معTwitter API ، ويرسل استعلام البحث، ويستقبل البيانات في دفعات، ثم يقوم بتخزينها مؤقتًا.

- **استخلاص البيانات من الويب(Web Scraping) :** عندما لا تتوفر واجهة API ، يتم اللجوء إلى تقنية Web Scraping، حيث يقوم المبرمج بكتابة شفرة تستخدم مكتبات قوية مثل BeautifulSoup أو Scrapyلـ "قراءة" هيكل صفحة الويب (HTML) واستخلاص المعلومات المحددة (مثل جداول البيانات، أو نصوص المقالات) وتجاهل العناصر غير الضرورية (مثل الإعلانات)، وهذه العملية تتطلب فهمًا عميقًا لهيكل DOM للصفحة لضمان أن الكود لا ينكسر عند تغيير تصميم الموقع،

مثال :استخلاص بيانات أسعار المنتجات من مواقع منافسة عن طريق محاكاة تصفح المستخدم وجمع الأرقام والنصوص ذات الصلة.

ثانيًا: التنظيف والمعالجة المسبقة(Preprocessing) : ضمان "صحة" البيانات

المرحلة الحاسمة التالية هي المعالجة المسبقة، وهي في جوهرها سلسلة من العمليات الرياضية والمنطقية المطبقة برمجيًا على البيانات الخام لجعلها صالحة للاستخدام بواسطة خوارزميات التعلم الآلي، وغالبًا ما تُستخدم مكتبة Pandasللتعامل مع هياكل البيانات الجدولية و NumPyللعمليات الرقمية عالية الأداء.

- **معالجة القيم المفقودة :(Handling Missing Values)** يتطلب هذا شفرات برمجية لاتخاذ قرارات منطقية بشأن كيفية التعامل مع الخلايا الفارغة، ويمكن أن يتضمن الكود إحدى الاستراتيجيات التالية:
  - ✓ **الإزالة(Dropping) :** حذف الصفوف التي تحتوي على قيم مفقودة (إذا كان عددها قليلاً).
  - ✓ **الاستعاضة(Imputation) :** ملء القيم المفقودة باستخدام قيمة حسابية (مثل **المتوسط** mean() أو **الوسيط** median()) للعمود المعني،
  - ✓ **الاستدلال(Inference) :** استخدام نموذج تعلم آلي آخر للتنبؤ بالقيم المفقودة.
  - ✓ **مثال:** إذا كانت لدينا بيانات مسح سكاني، ووُجدت قيم مفقودة في عمود "الدخل"، فستقوم الشفرة بحساب متوسط الدخل لأشخاص بنفس الفئة العمرية والتعليمية واستخدام هذا المتوسط لملء الفراغات.

- **توحيد الصيغ وتحويل البيانات:** يجب أن تقوم الشفرة بفرض التناسق عبر مجموعة البيانات، وهذا يشمل تحويل الحروف النصية إلى حالة واحدة (لضمان أن "Apple" لا تُعامل ككلمة مختلفة عن"apple")، وتحويل أنواع البيانات (مثل تحويل سلاسل نصية تمثل أرقامًا إلى أرقام حقيقية)، وتصحيح الأخطاء الإملائية،
  - ✓ **مثال:** تحويل تواريخ مكتوبة بصيغ مختلفة ("01/10/2025"، "01-أكتوبر-2025") إلى صيغة موحدةISO 8601 .

- **تحويل البيانات النصية إلى متجهات رقمية(Vectorization) :** الخوارزميات الرياضية لا تفهم الكلمات، لذا، تتطلب البرمجة تحويل النصوص إلى تمثيلات رقمية تُعرف باسم المتجهات(Vectors) ، وتُستخدم هنا تقنيات مثل حقيبة الكلمات (Bag-of-Words) أو TF-IDFأو تضمين الكلمات (Word Embeddings) مثل Word2Vec أوBERT ، ويقوم المبرمج بكتابة الكود اللازم لتطبيق هذه النماذج الرياضية المعقدة لإنشاء مصفوفات رقمية ضخمة تمثل النصوص.

مثال: استخدام مكتبة Scikit-learn لتطبيق TfidfVectorizer لتحويل مجموعة من مراجعات العملاء النصية إلى مصفوفة رقمية يمكن لنموذج التصنيف (Classification) معالجتها.

- **تطبيع البيانات وتوحيد مقياسها(Normalization and Scaling) :** لكي تعمل الخوارزميات بكفاءة وتصل إلى التقارب (Convergence) بشكل أسرع، يجب أن تكون جميع الميزات (Features) ضمن نطاق قياسي، وتتضمن البرمجة هنا تطبيق معادلات رياضية (مثل Z-score normalizationأو Min-Max scaling) على كل عمود في البيانات،

باستخدام مكتبات مثل Scikit-learn's StandardScaler ، وهذا يضمن عدم هيمنة ميزة واحدة على حساب الميزات الأخرى لمجرد أن مقياسها أكبر (مثل الدخل مقابل العمر).

تتطلب هندسة البيانات كتابة شفرات برمجية تتجاوز مجرد التنفيذ، بل يجب أن تكون قابلة للتوسع (Scalable) للتعامل مع بيتابايت من البيانات و قابلة للصيانة (Maintainable) لضمان استمرارية عمل النموذج لسنوات.

ب. تطوير الخوارزميات وتطبيقها(Algorithm Implementation) : ترجمة المنطق الرياضي إلى كود

إذا كانت هندسة البيانات هي بناء الأساس، فإن تطوير الخوارزميات وتطبيقها هو بناء هيكل الذكاء الاصطناعي نفسه، وهذا هو المكان الذي يُترجم فيه المنطق الرياضي المعقد (الذي يشكل جوهر الذكاء الاصطناعي، مثل خوارزمية الانتشار الخلفي Backpropagation - التي تُستخدم لتحديث الأوزان) إلى شفرة قابلة للتنفيذ على معالجات الكمبيوتر (CPUs)أو وحدات معالجة الرسوميات(GPUs) ، وتتطلب هذه المرحلة فهمًا عميقًا للرياضيات والبرمجة معًا، وتتم غالبًا باستخدام أطر عمل مثل TensorFlowأو PyTorch.

أولا: بناء الشبكات العصبونية(NN Architecture) : تعريف الهيكل الدقيق

تُعد الشبكة العصبونية (Neural Network) هي الهيكل الأساسي لمعظم نماذج الذكاء الاصطناعي الحديثة، وتتضمن البرمجة في هذه المرحلة كتابة شفرة لـ تحديد الهيكل (Architecture) الشبكي بشكل حرفي، طبقة تلو الأخرى، مع تحديد دقيق لجميع المعلمات:

- **تحديد الطبقات(Defining Layers) :** يجب على المبرمج كتابة شفرة لتحديد نوع كل طبقة (مثل طبقة كثيفة الاتصال Dense/Fully Connected Layer، أو طبقة الالتفاف Convolutional Layerفي حالة معالجة الصور، أو طبقة التكرار Recurrent Layer في حالة معالجة النصوص المتسلسلة)، ويتطلب الكود تحديد عدد العقد (Neurons) في كل طبقة، وهو ما يحدد سعة النموذج وقدرته على التعلم.

مثال (باستخدامKeras/TensorFlow): يُترجم تصميم نموذج لشبكة عصبونية بسيطة إلى كود يبدو كالتالي:

```
model = Sequential([
    Dense(128, activation='relu', input_shape=(input_features,)), # الطبقة الأولى
    Dropout(0.2), # طبقة إسقاط
    Dense(64, activation='relu'),
    Dense(10, activation='softmax') # طبقة الخرج
])
```

وهذا الكود يقوم بـ ترجمة الفكرة النظرية لشبكة عصبونية متعددة الطبقات إلى كائن رقمي يمكن تدريبه.

- **تحديد دوال التنشيط(Activation Functions) :** تلعب دوال التنشيط (مثل **ReLU**، **Sigmoid**، أو **Softmax**) دورًا حيويًا في إدخال **اللاخطية (Non-linearity)** إلى النموذج، مما يمكنه من تعلم علاقات أكثر تعقيدًا في البيانات، وتتطلب البرمجة تحديد الدالة المناسبة لكل طبقة بناءً على طبيعة المهمة، وعلى سبيل المثال، تُستخدم Softmaxغالبًا في الطبقة الأخيرة لمهام التصنيف متعدد الفئات، حيث تُحوّل مخرجات النموذج إلى احتمالات مجموعها يساوي 1.

مثال: في نموذج تصنيف صور لـ 10 أنواع من الحيوانات، يجب على المبرمج تحديد 10 عقد في طبقة الخرج واستخدام دالة Softmax، لضمان أن التنبؤات النهائية هي احتمالات متسقة.

- **إدارة الأوزان والانحيازات(Weights and Biases) :** على الرغم من أن إطار العمل (مثلPyTorch) يدير التهيئات الأولية (Initialization) بشكل آلي، إلا أن المبرمج قد يحتاج في بعض الأحيان إلى برمجة طرق تهيئة متقدمة (مثل He Initializationأو Xavier Initialization) لضمان بدء عملية التدريب بشكل مستقر، وهذا يتطلب كتابة شفرة برمجية تؤثر مباشرة على القيم الأولية لمصفوفات الأوزان.

ثانيًا: تحديد الخسارة والمحسِّن(Loss and Optimizer) : توجيه عملية التعلم

يجب أن يتم تزويد النموذج بأدوات رياضية لتوجيه عملية التعلم، وهذا يتم عبر دالة الخسارة (Loss Function) وخوارزمية المحسِّن(Optimizer) ، وتتضمن البرمجة هنا "تجميع" المكونات الرياضية معًا وتحديد كيف سيتفاعل النموذج معها.

- **برمجة دالة الخسارة(Loss Function) :** دالة الخسارة (تسمى أيضًا دالة التكلفة (Cost Function - هي مقياس رياضي يتم برمجته ليقوم بحساب مدى "سوء" أداء النموذج مقارنة بالنتائج الفعلية، ويجب أن يختار المبرمج الدالة الصحيحة للمهمة:

- ✓ للتصنيف(Classification) : تُستخدم) Cross-Entropyلأنها تتوافق جيدًا مع مخرجات الاحتمالات).
- ✓ للانحدار(Regression) : يُستخدم) Mean Squared Error (MSE)لأنه يقيس متوسط تربيع الفروقات).
- ✓ ويتطلب الأمر كتابة شفرة تُحدد هذه الدالة وتجعلها جزءًا من عملية تدريب النموذج،
- ✓ **مثال:** في مشروع التنبؤ بسعر منزل (مشكلة انحدار)، يتم برمجة دالة MSE حيث تكون الخسارة هي:

$$loss = \left({}_i y + \hat{}_i y\right)_i = 1^n$$

ويتم حساب هذه القيمة برمجيًا بعد كل دفعة من البيانات(Batch) .

- **برمجة خوارزميات المحسِّن(Optimizer) :** المُحسِّن هو الخوارزمية التي تستخدم قيمة الخسارة المحسوبة (عبر تقنية الانتشار الخلفي (Backpropagation - لتحديد كيفية تعديل أوزان النموذج، ووظيفة المبرمج هي اختيار المحسِّن الأنسب وتحديد معلّماته الفائقة(Hyperparameters) ،

- ✓ **المُحسِّنات الشائعة:** RMSprop (Stochastic Gradient Descent), SGD, Adam .
- ✓ **التكويد(Coding) :** يتطلب الأمر سطرًا برمجيًا لـ "تجميع (Compile) "النموذج وتحديد المحسِّن ومعدل التعلم (Learning Rate)الخاص به،

مثال:

```
model.compile(
    optimizer=Adam(learning_rate=0.001), # تحديد خوارزمية Adam وتحديد معدل التعلم
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

وهذا الكود يقوم بترجمة القرار الرياضي باستخدام المشتقات الجزئية لتحديث الأوزان إلى خطوة تنفيذية للنموذج، وتعتبر هذه المرحلة هي النقطة التي يلتقي فيها العلم الرياضي بالهندسة البرمجية، حيث يجب على المبرمج التأكد من أن جميع المكونات الرياضية تعمل معًا بكفاءة لتمكين النموذج من تحقيق **خاصية العمومية** عبر التعلم الفعال من البيانات.

ج. النشر والصيانة :(Deployment and Maintenance) تحويل النموذج إلى قيمة عملية

النماذج المدربة (Trained Models) هي مجرد ملفات رقمية على قرص صلب ما لم يتم دمجها في تطبيقات عملية يمكن للمستخدمين النهائيين التفاعل معها، ويمثل النشر (Deployment) مرحلة التحول من المختبر إلى الحياة الواقعية، وتتطلب هذه المرحلة مهارات برمجية عالية في تطوير الواجهات، وأنظمة التشغيل، والأتمتة لضمان استمرار عمل النموذج بكفاءة وموثوقية، وهي المرحلة التي تُحول نتائج الذكاء الاصطناعي إلى تطبيقات رقمية عملية ومفيدة.

أولا: بناء واجهة برمجية(API Development) : جسر التواصل الفوري

لتحويل النموذج المدرب إلى خدمة، يجب بناء واجهة برمجية للتطبيق (API) تتيح للتطبيقات الأخرى (مثل تطبيقات الهواتف الذكية، مواقع الويب، أو الأنظمة الخلفية للشركات) إرسال البيانات إلى النموذج واستقبال التنبؤات (Predictions)منه في الوقت الفعلي(Real-time) ، وغالبًا ما يتم استخدام أطر عمل برمجية مثل Flaskأو FastAPIفي Pythonلهذه المهمة.

- ✓ **إنشاء نقطة نهاية(Endpoint) :** تتطلب البرمجة هنا كتابة شفرة لإنشاء نقطة نهاية (مثل /predictأو /classify) يمكن الوصول إليها عبر طلبات HTTP (POST)، ويجب أن يتضمن الكود:
- ✓ **تحميل النموذج:** تحميل ملف النموذج المدرب (مثل ملف .pkl أو .h5) في الذاكرة لتقليل زمن الاستجابة،
- ✓ **معالجة المدخلات:** استقبال البيانات الواردة (غالبًا بصيغةJSON) ، والقيام بعمليات المعالجة المسبقة الضرورية (Preprocessing) عليها لضمان أنها بنفس التنسيق الذي يتوقعه النموذج،
- ✓ **التنبؤ:** استدعاء دالة التنبؤ (مثل model.predict()) على البيانات،
- ✓ **تنسيق المخرجات:** تحويل مخرجات النموذج الرقمية إلى تنسيق يمكن للبشر قراءته (مثل رسالة JSONوإرسالها مرة أخرى إلى التطبيق الطالب،

مثال: في نظام تشخيص طبي يعتمد على الذكاء الاصطناعي، يرسل تطبيق الهاتف صورة الأشعة عبر الAPI ، ويقوم الكود الخلفي بتحويل الصورة إلى مصفوفة رقمية(Preprocessing) ، ويغذيها للنموذج، ثم يعيد النتيجة ({"diagnosis" "Benign", "confidence": 0.95})، وهذا كله يتم في أقل من ثانية واحدة.

- ✓ **التهيئة والحاويات(Containerization) :** لضمان أن يعمل الـAPI بشكل متطابق في جميع البيئات (بيئة المطور، بيئة الاختبار، بيئة الإنتاج)، يتم برمجة ملفات تهيئة (مثل ملفات Dockerfile)لإنشاء حاويات (Containers) باستخدام Docker، وهذا يضمن حزم شفرة النموذج، والAPI ، وجميع مكتبات Python المطلوبة في حزمة واحدة موحدة.

ثانيًا: المراقبة والأتمتة :(Monitoring and Automation) ضمان الكفاءة والعمومية

النشر ليس نهاية القصة، فالنماذج تخضع لما يُعرف بـ انحراف النموذج(Model Drift) ، حيث يتدهور أداؤها بمرور الوقت بسبب تغير طبيعة البيانات الواقعية(Real-world Data) ، ويتطلب ذلك برمجة مستمرة لأنظمة المراقبة والأتمتة.

- **المراقبة المستمرة للأداء(Performance Monitoring) :** يجب كتابة شفرات برمجية تعمل على مدار الساعة للتحقق من **مؤشرات الأداء الرئيسية (KPIs)** للنموذج، وهذا يتضمن:

✓ **مراقبة زمن الاستجابة(Latency)** : التأكد من أن النموذج لا يستغرق وقتًا طويلاً في إرسال التنبؤات،

✓ **مراقبة جودة التنبؤ(Prediction Quality)** : التحقق من دقة التنبؤات الفعلية (إذا كان هناك بيانات حقيقة متاحة لاحقًا).

✓ **اكتشاف انحراف البيانات(Data Drift Detection)** : برمجة خوارزميات إحصائية (مثل اختبار Kolmogorov-Smirnov) للمقارنة بين توزيع البيانات الجديدة الواردة وتوزيع بيانات التدريب الأصلية،

مثال: إذا كان نموذج التنبؤ بالاحتيال يتلقى فجأة بيانات مالية بقيم أعلى بكثير مما تدرب عليه، فإن شفرة المراقبة يجب أن تطلق إنذارًا آليًا (Alert) للمهندس.

- **أتمتة عملية إعادة التدريب(Retraining Automation)** : لضمان خاصية العمومية للنموذج على المدى الطويل، يجب إعادة تدريبه بشكل دوري على مجموعة بيانات جديدة، ويتطلب ذلك كتابة شفرات أتمتة (Automation Scripts)تُدعى خطوط أنابيب تعلم الآلة(MLOps Pipelines) .

✓ **آلية العمل:** يقوم الكود البرمجي بجدولة (Scheduling) عملية إعادة تدريب النموذج بالكامل (تبدأ من الاستخلاص، ومرورًا بالتنظيف، وصولًا إلى التدريب والنشر)، ويتم ذلك بشكل آلي عندما تشير بيانات المراقبة إلى تدهور في الأداء أو انحراف كبير في البيانات.

مثال: استخدام أدوات مثل **Airflow**أو **Kubeflow** لبرمجة سير عمل (Workflow) يتم تنفيذه شهريًا: 1) سحب البيانات الجديدة، تنظيفها، تدريب النموذج، اختبار النموذج الجديد، ثم نشر النموذج الجديد واستبدال القديم في الAPI .

والبرمجة في مرحلة النشر والصيانة هي التي تضمن أن استثمار الشركة في بناء نموذج الذكاء الاصطناعي يستمر في تقديم قيمة عمل حقيقية في بيئة ديناميكية ومتغيرة باستمرار.

2. مجالات البرمجة التي تدعم أنظمة الذكاء الاصطناعي

مجالات البرمجة ليست متجانسة؛ فكل مجال يساهم في دعم الذكاء الاصطناعي بطريقة مختلفة، مما يضمن أن يكون المنتج النهائي نظاماً متكاملاً وفعالاً.

مجال البرمجة	دورها في دعم الذكاء الاصطناعي	الخصائص البرمجية الأساسية	أمثلة تطبيقية في الذكاء الاصطناعي
برمجة تحليل البيانات (Data Science Programming)	تنظيف، تحليل، وتصوير البيانات الضخمة، واختيار الميزات (Feature Engineering).	استخدام الهياكل البيانية المتقدمة (Matrices, DataFrames) ومبادئ **الفاعلية**.	تحديد الأنماط في بيانات الرأي العام قبل تدريب نموذجNLP .
برمجة الواجهة الخلفية (Backend Programming)	إدارة قواعد البيانات، وتوفير البنية التحتية، وتلقي طلبات النماذج وإرسال الردود.	التعامل مع الخوادم(Servers) ، الأمن، وضمان **التسلسل** المنطقي للبيانات.	بناء خادم يرسل صورة إلى نموذج التعرف على الوجه ويسترجع نتيجة المصادقة.
برمجة الأنظمة المدمجة (Embedded Programming)	تزويد الروبوتات والعتاد المادي (Hardware)بالمنطق اللازم للتفاعل مع العالم.	لغات منخفضة المستوى (مثل C) والتحكم الدقيق في الذاكرة لضمان **السرعة**.	برمجة نظام الفرملة الطارئة في السيارة ذاتية القيادة
برمجة الواجهة الأمامية (Frontend Programming)	بناء واجهات المستخدم التي تتفاعل مع مخرجات الذكاء الاصطناعي.	لغات تصميم الواجهات (HTML, CSS, JavaScript)لتقديم **المخرجات** للمستخدم.	عرض توقعات الطقس أو توصيات الأفلام بشكل رسومي جذاب.

3. العملية المنهجية لدمج البرمجة في الذكاء الاصطناعي(AI Development Lifecycle)

يجب على الطالب فهم أن تطوير نظام ذكي ليس مجرد كتابة كود، بل هو عملية منهجية ومكررة (Iterative Process) تتطلب مهارات برمجية في كل مرحلة.

المرحلة الأولى: تعريف المشكلة والنمذجة الرياضية

- **المهارة البرمجية:** القدرة على تحويل المشكلة الواقعية (مثل "التنبؤ بأزمة") إلى مقياس كمّي وهدف رياضي (مثل "تقليل نسبة الخطأ .("**Loss**

المرحلة الثانية: جمع البيانات وإعدادها(Preparation)

- **الدور البرمجي:** استخدام لغات عالية المستوى (Python) لتنظيف وتوحيد البيانات. هذه هي المرحلة التي تُطبق فيها معظم مهارات **هندسة البيانات** المذكورة سابقاً.

المرحلة الثالثة: بناء النموذج والتدريب(Modeling & Training)

- **الدور البرمجي:** كتابة كود الشبكة العصبونية (باستخدام أطر مثل TensorFlow وتعيين المتغيرات، وضبط معاملات التعلم (Hyperparameters) بشكل دقيق، وهذه العملية تتطلب تطبيقًا صارمًا لخاصية الدقة والمنطق البرمجي.

المرحلة الرابعة: التقييم والتحقق(Evaluation)

- **الدور البرمجي:** كتابة دوال برمجية لحساب مقاييس الأداء (Metrics) مثل الدقة(Accuracy) ، الاستدعاء (Recall)، ومقياس F1 Score، وتستخدم هذه المقاييس لتحديد مدى صحة (Correctness) النموذج.

المرحلة الخامسة: النشر(Deployment)

- **الدور البرمجي:** كتابة كود الواجهة الخلفية (Backend) لتغليف النموذج في خدمة جاهزة للاستخدام(API) ، وضمان أن النظام يلبي متطلبات **الفاعلية**)السرعة والذاكرة) في البيئة التشغيلية.

مقارنة الدور	البرمجة التقليدية (حل مشكلة)	البرمجة لدعم الذكاء الاصطناعي (بناء نموذج)
أولوية الكود	التركيز على **منطق العمل** (Business Logic) والتسلسل الواضح.	التركيز على **معالجة البيانات الضخمة** والكفاءة في التدريب.
النتيجة	ناتج محدد ومضمون (مثل **الخوارزمية** في الآلة الحاسبة).	ناتج **احتمالي** وتقديري (مثل **التنبؤ** بأسعار الأسهم).
المهارة المطلوبة	إتقان بناء الجملة (Syntax) وبنى التحكم.	إتقان المكتبات الرياضية والإحصائية (NumPy, Pandas, Scikit-learn).

تمارين وتطبيقات عملية (ربط المنطق البرمجي بالذكاء الاصطناعي)

التمرين 1: دور المعالجة المسبقة للبيانات(Preprocessing)

لنفترض أنك تعمل على نموذج للتعرف على المشاعر في التعليقات المكتوبة(NLP) ، وقبل إدخال التعليق (مثل "المنتج رائع لكن بطيء")، يجب على البرمجة أن تعده:

1. **المشكلة:** النص "المنتج رائع لكن بطيء" يحتوي على كلمتين متضادتين.
2. **الدور البرمجي:** استخدام دالة برمجية لتقسيم الجملة إلى كلمات (Tokenization) ثم تحويل الكلمات إلى أرقام (Vectorization).

3. **لماذا البرمجة ضرورية؟** لأن نموذج الذكاء الاصطناعي (الذي يعمل على المصفوفات والأرقام) لا يمكنه قراءة النص مباشرةً؛ بل يحتاج إلى **تسلسل** من عمليات التحويل البرمجية المنهجية لترجمة المدخل البشري إلى شكل يمكن معالجته.

التمرين 2: تطبيق خاصية الفاعلية (Efficiency) في النشر

إذا كان نموذجك للذكاء الاصطناعي يستغرق 10 ثوانٍ لتقديم استجابة واحدة:

1. **السؤال:** ما هي الخاصية البرمجية التي تم إهمالها، ولماذا تعتبر خطراً في أنظمة التوصية على منصات البث؟
2. **الإجابة:** خاصية الفاعلية(Efficiency) ، وتحديداً التعقيد الزمني، وهذا خطر لأنه سيجعل المستخدم ينتظر 10 ثوانٍ بعد كل نقرة للحصول على توصية، مما يدمر تجربة المستخدم ويقلل من الإنتاجية.

التمرين 3: تحديد وظيفة البرمجة في سيناريو الأزمة

في سيناريو إدارة أزمة ما، يُطلب منك جمع بيانات ضخمة من تويتر لتحليل الرأي العام. ما هو الدور البرمجي الحاسم في هذه اللحظة؟

1. **الدور: هندسة البيانات.(Data Engineering)**
2. **الوظيفة:** كتابة شفرة برمجية للاتصال بواجهة برمجة تطبيقات تويتر (API) بشكل مستمر، وتجميع البيانات في الوقت الفعلي، وتخزينها في قاعدة بيانات منظمة قبل إرسالها إلى نموذج التحليل.