الفكرة الأساسية هي أن الشبكة العصبية ستتعلم التمييز بين "التحميص الجيد" (Good Roast) و"التحميص السيء" (Bad Roast) بناءً على متغيرين فقط: **درجة الحرارة** (Temperature) بالدرجة المئوية، و**المدة** (Duration) بالدقائق.

الخطوات:

1. **تجهيز البيانات (Data Processing):**"توحيد المعايير" (Normalization) للبيانات قبل إدخالها للشبكة.
2. **هيكلية النموذج (Model Architecture):** تصميم الطبقات (Layers) و دور دالة التنشيط (Sigmoid) في اتخاذ القرار.
3. **التدريب والتنبؤ (Training & Predictions):** يقوم النموذج بتعديل "الأوزان" (Weights) لتقليل الخطأ ونحسب القرار النهائي.

1. تجهيز / تسوية البيانات (Normalization) 📊

قبل أن تبدأ الشبكة العصبية في التعلم، علينا أن ننظر إلى البيانات التي ستدخل إليها. لدينا ميزتان (Features) تحددان جودة التحميص:

1. **درجة الحرارة (Temperature):** تتراوح تقريبًا بين 151 و 285 درجة مئوية.
2. **المدة (Duration):** تتراوح تقريبًا بين 11.5 و 15.5 دقيقة.

لاحظ الفرق الكبير في الأرقام: المئات (مثل 200) مقابل العشرات (مثل 12).
السبب يكمن في **الرياضيات** التي تجري خلف الكواليس. الشبكة العصبية تقوم بضرب "الوزن" (w) في "المدخل" (x).

- بما أن درجة الحرارة تصل إلى **200** والمدة حوالي **12**، فإن أي تغيير بسيط جدًا في "وزن" الحرارة سيؤدي إلى قفزة ضخمة في النتيجة النهائية مقارنة بنفس التغيير في وزن المدة.
- هذا الاختلاف الكبير يجعل "ساحة التعلم" (أو دالة التكلفة) تبدو وكأنها وادٍ ضيق وطويل جدًا. نتيجة لذلك، تتعثر خوارزمية التعلم وتتأرجح يمينًا ويسارًا ببطء شديد بدلاً من النزول بسلاسة نحو الحل الأمثل.

لحل هذه المشكلة، نقوم بـ **التسوية (Normalization)**، وهي عملية ضغط وتعديل للأرقام لتصبح الميزتان (الحرارة والمدة) في نطاق رقمي متشابه (عادة ما يكون قريبًا من -1 إلى 1).

الآن، لنطبق هذا عمليًا من الملف. في الصفحة رقم 2 (السطر 57)، استخدمنا طبقة Normalization.

ثم **التكييف (Adaptation)**:

- **الأمر:**

```
norm_1.adapt(X)
```

- **الوظيفة:** تقوم طبقة التوحيد (Normalization) بتعلم **متوسط** (Mean) و **تباين** (Variance) كل ميزة من البيانات الأصلية (X). هذا يسمح لها بتحويل جميع المدخلات إلى نطاق رقمي مماثل.
- **النتيجة:** بعد التوحيد، أصبحت قيم المدخلات في نطاق متقارب، حيث أقصى وأدنى قيمة تقتربان من 1.7 و -1.7 تقريبًا (بدلاً من المئات).

بعد ذلك، تم نسخ البيانات الموحدة (Xn) ألف مرة باستخدام np.tile. هذا نتج عنه مجموعة تدريب كبيرة جداً بحجم 200,000 مثال ، والهدف من ذلك هو **تقليل عدد "العصور" (Epochs)** المطلوبة للتدريب.

2. هيكلية نموذج TensorFlow 🏗️

ننتقل الآن إلى بناء "شبكة تحميص القهوة" (Coffee Roasting Network) في قسم 1.2 من المعمل. تم إنشاء النموذج باستخدام tf.keras.models.Sequential:

- **مدخلات النموذج:** تحدد أولاً بحجم (2,) لتمثيل الميزتين (درجة الحرارة والمدة).
- **الطبقة الأولى (Layer 1):**

```
Dense(3, activation='sigmoid', name='layer1')
```

  - **عدد الوحدات:** 3 خلايا عصبية (Neurons).
  - **دالة التنشيط: Sigmoid**، التي تخرج قيماً بين 0 و 1.
  - **عدد المعاملات (Parameters):** 9 معاملات. تُحسب كالتالي: (2 مدخلات *3 وحدات) + 3 انحيازات= 9.

- **الطبقة الثانية (Layer 2 - طبقة المخرجات):**

```
Dense(1, activation='sigmoid', name='layer2')
```

  - **عدد الوحدات:** 1 وحدة (تمثل الاحتمالية النهائية للتحميص الجيد).
  - **دالة التنشيط: Sigmoid**.
  - **عدد المعاملات (Parameters):** 4 معاملات. تُحسب كالتالي: (3 مدخلات *1 وحدة)+ 1 انحياز = 4.

إجمالي المعاملات القابلة للتدريب في الشبكة هو 13.

3. تجميع النموذج وتجهيزه للتدريب (Model Compilation) 🛠️

الخطوة التالية في المعمل هي **تجميع النموذج (Compiling the Model)** باستخدام أمر `model.compile`. هذه الخطوة أساسية، حيث تخبر الشبكة "كيف" يجب أن تتعلم.

هنا نحدد أهم مكونين لعملية التعلم:

1. **دالة الخسارة (Loss Function):**

```
loss=tf.keras.losses.BinaryCrossentropy()
```

   - **وظيفتها:** تقيس دالة الخسارة مدى **سوء** أداء النموذج. إنها تحسب الفرق بين النتيجة المتوقعة للنموذج (احتمالية التحميص الجيد، وهي قيمة بين 0 و 1) والقيمة الحقيقية (0 للسيئ، 1 للجيد).
   - **لماذا Binary Cross-Entropy؟** لأن مشكلتنا هي **تصنيف ثنائي (Binary Classification)**: لدينا فئتان فقط (تحميص جيد أو سيئ). هذه الدالة هي الأنسب لقياس جودة التوقعات الاحتمالية في هذا النوع من المشاكل.

2. **المُحسّن (Optimizer):**

```
optimizer=tf.keras.optimizers.Adam(learning_rate=0.01)
```

   - **وظيفته:** هذا هو المحرك الذي يقوم بتعديل **أوزان** الشبكة (W) و **انحيازاتها** (b) في كل خطوة تدريب.
   - **كيف يعمل؟** يستخدم المُحسّن تقنية تسمى "الانحدار التدرجي" (Gradient Descent). تخيل أنك تحاول النزول إلى قاع وادٍ (الذي يمثل أقل خسارة)، المُحسّن يحدد الاتجاه والخطوة المناسبة لتعديل الأوزان بأفضل طريقة ممكنة.

---

4. تدريب النموذج (Model Training) 🏃

بعد التجميع، تأتي خطوة **التدريب (Fitting the Model)** باستخدام أمر `model.fit`.

- **الأمر:**

```
model.fit(Xn, Y, epochs=10)
```

  - **المدخلات (Xn):** البيانات التي تم توحيدها وتكرارها (200,000 مثال).
  - **المخرجات/التسميات (Y):** التسميات الحقيقية (0 أو 1).
  - **العصور (epochs=10):** تعني أن النموذج سيمر على مجموعة التدريب الكاملة **10 مرات**.

خلال التدريب، ينخفض رقم **الخسارة (Loss)** بشكل تدريجي مع كل عصر. هذا الانخفاض يدل على أن النموذج أصبح "أذكى" وأن أوزانه الجديدة تتنبأ بالنتيجة الصحيحة (جيد/سيئ) باحتمالية أعلى.

5. التنبؤ بالنتائج (Making Predictions) ☕

بعد الانتهاء من تدريب النموذج لـ 10 عصور (epochs) وانخفاض دالة الخسارة، يصبح النموذج جاهزًا للاستخدام. الهدف الآن هو إدخال أي تركيبة جديدة من **درجة الحرارة** و**المدة** والحصول على تنبؤ بجودة التحميص.

الأمر المستخدم:

```
model.predict(X_new)
```

- **المدخلات:** يجب أن تكون المدخلات الجديدة (X_new) **موحدة** (Normalized) بنفس الطريقة التي تم بها توحيد بيانات التدريب، وذلك باستخدام طبقة norm_1. هذا يضمن أن النموذج يرى الأرقام في النطاق الذي تدرب عليه.
- **المخرجات:** يخرج النموذج **احتمالية** (Probability) تتراوح بين **0** و **1**.
  - القيمة القريبة من **1** تشير إلى احتمالية عالية بأن التحميص **جيد**.
  - القيمة القريبة من **0** تشير إلى احتمالية عالية بأن التحميص **سيئ**.

مثال تطبيقي من المعمل:
إذا أدخلت للنموذج بيانات غير موحدة مثل:

- درجة حرارة 200°C، ومدة 13.5 دقيقة.

سيقوم النموذج أولاً بتوحيدها، ثم يمررها عبر الشبكة، ليخرج لك احتمالية مثل 0.98، مما يعني أن النموذج يتوقع بنسبة 98% أن يكون هذا **تحميصًا جيدًا**.

---

6. تحليل وتصور القرار (Analysis and Visualization) 📈

للتأكد من أن النموذج تعلم بشكل صحيح، فإن أهم خطوة هي **تصور (Visualization)** منطقه في اتخاذ القرار على الرسم البياني، وهو ما يتم في القسم الأخير من المعمل.

أ. خريطة الاحتمالية (Network Probability Map)

يقوم المعمل برسم خريطة للنتائج (انظر الرسم البياني الأيسر في الصفحة الأخيرة من الملف).

1. **المحاور:** تمثل المحاور **درجة الحرارة** (الأفقية) و**المدة** (العمودية).
2. **التظليل الأزرق (Shading):** يمثل اللون الأزرق **مخرج الطبقة النهائية** (الاحتمالية).
   - **الأزرق الغامق:** يمثل احتمالية عالية جدًا (قريبة من 1)، أي منطقة يتوقع فيها النموذج "تحميصًا جيدًا".
   - **الأزرق الفاتح/الأبيض:** يمثل احتمالية منخفضة (قريبة من 0)، أي منطقة يتوقع فيها النموذج "تحميصًا سيئًا".

3. **تفسير النتيجة:** يوضح هذا الرسم كيف تعلم النموذج أن منطقة معينة في مركز البيانات، حيث تكون درجة الحرارة والمدة ضمن المدى المثالي، هي المنطقة ذات الاحتمالية العالية للتحميص الجيد، بينما المناطق البعيدة عنها تكون ذات احتمالية منخفضة.

ب. حدود القرار النهائية (Network Decision Boundary) 🎯

الخطوة الأخيرة هي تحويل الاحتمالية إلى **قرار ثنائي نهائي** (Good Roast / Bad Roast). هذا موضح في الرسم البياني الأيمن.

1. **عتبة القرار (Threshold):** يتم تطبيق قاعدة بسيطة: **إذا كانت الاحتمالية ≥ 0.5، فالقرار هو "تحميص جيد" (Predicted Good Roast).**
2. **المنطقة المحددة:** ينتج عن هذه العتبة خط (أو منحنى) يفصل بين منطقتين واضحتين على الرسم البياني:
    - **منطقة 'Good Roast' (O):** وهي المنطقة التي يقرر فيها النموذج أن التحميص جيد.
    - **منطقة 'Bad Roast' (X):** وهي المنطقة التي يقرر فيها النموذج أن التحميص سيئ.