Comenzado el	miércoles, 4 de junio de 2025, 11:06
Estado	Finalizado
Finalizado en	miércoles, 4 de junio de 2025, 11:52
Tiempo empleado	45 minutos 47 segundos
Calificación	**23,33** de 30,00 (**78**%)

Navegación por el cuestionario

1	2	3	4	5	6	7
✓	🟥	✓	✓	✓	✓	🟥
8	9	10				
▨	✓	✓				

Mostrar una página cada vez
Finalizar revisión

Pregunta **1**
Correcta
Puntúa 3,33 sobre 3,33
Marcar pregunta

Considere el siguiente algoritmo dado en la forma escalar:

```
sub dataset  minmax{
 my ($self, $dataset) = @_;
 my @minmax;
 for my $i (0 .. $#{$dataset->[0]}){
  my $col_values = [map {$_->[$i]}  @$dataset];
  my $value_min = min(@$col_values);
  my $value_max = max(@$col_values);
  push @minmax, [$value_min, $value_max];
 }
 return \@minmax;
}
```

Elija la opción que correctamente convierte el código escalar a la forma tensorial equivalente, la cual arroja los mismos resultados.

Seleccione una:

● a.
```
sub dataset_minmax {
 my ($self, $dataset) = @_;
 return mx->nd->stack($dataset->min(axis => 0), $dataset->max(axis => 0), axis => 1);
}
```
✓

Correcta porque:

- `axis => 0` calcula mínimo y máximo **por columna**.
- `stack(..., axis => 1)` apila los valores de min y max en columnas.
- Resultado: un tensor de forma `[n_columns, 2]`, donde cada fila representa `[min, max]` de una columna.

○ b.
```
sub dataset_minmax {
 my ($self, $dataset) = @_;
 return mx->nd->stack($dataset->min(axis => 1), $dataset->max(axis => 1), axis => 1);
}
```

○ c.
```
sub dataset_minmax {
 my ($self, $dataset) = @_;
 return mx->nd->stack($dataset->min, $dataset->max, axis => 1);
}
```

○ d.
```
sub dataset_minmax {
 my ($self, $dataset) = @_;
 return mx->nd->concat($dataset->min(axis => 1), $dataset->max(axis => 1), dim => 0);
}
```

○ e.
```
sub dataset_minmax {
 my ($self, $dataset) = @_;
 return mx->nd->stack($dataset->min(axis => 1), $dataset->max(axis => 1), axis => 0);
}
```

Respuesta correcta

La respuesta correcta es:
```
sub dataset_minmax {
 my ($self, $dataset) = @_;
 return mx->nd->stack($dataset->min(axis => 0), $dataset->max(axis => 0), axis => 1);
}
```

Pregunta **2**
Incorrecta
Puntúa 0,00 sobre 3,33
Marcar pregunta

Considere el siguiente algoritmo dado en la forma escalar:

```
sub normalize_dataset{
 my ($self, $dataset, $minmax) = @_;
 for my $row (@$dataset){
  for my $pos (0 .. $#{$row}){
   $row->[$pos] = ($row->[$pos] - $minmax->[$pos][0]) / ($minmax->[$pos][1] - $minmax->[$pos][0]);
  }
 }
}
```

Elija la opción que correctamente convierte el código escalar a la forma tensorial equivalente, la cual arroja los mismos resultados.

Seleccione una:

● a.
```
sub normalize_dataset {
 my ($self, $dataset, $minmax) = @_;
 my $min = $minmax->slice_axis(axis => 1, begin => 0, end => 1);
 my $max = $minmax->slice_axis(axis => 1, begin => 1, end => 2);
 $dataset->slice( [0, $dataset->shape->[0] -1], [0, $dataset->shape->[1] -1]) .= ($dataset - $min) / ($max - $min);
}
```
✖

❌ Por qué es incorrecta:

- Aunque el slicing de columnas es correcto (`axis => 1`), **falta la transposición (`->T`)** de `$min` y `$max`.
- Como resultado, `$min` y `$max` tienen forma `(n_features, 1)` en lugar de `(1, n_features)`, lo que causa **incompatibilidad de forma** durante la operación vectorizada con `$dataset` de forma `(n_samples, n_features)`.

○ b.
```
sub normalize_dataset{
 my ($self, $dataset, $minmax) = @_;


 my $min = $minmax->slice_axis(axis=>1, begin=>0, end=>1)->T;
 my $max = $minmax->slice_axis(axis=>1, begin=>1, end=>2)->T;
 $dataset->slice( [0, $dataset->shape->[0] -1], [0, $dataset->shape->[1] -1]) .= ($dataset - $min) / ($max - $min);

}
```

○ c.
```
sub normalize_dataset {
 my ($self, $dataset, $minmax) = @_;
 my ($min, $max) = @{ $minmax };
 $dataset->slice( [0, $dataset->shape->[0] -1], [0, $dataset->shape->[1] -1]) .= ($dataset - $min) / ($max - $min);
}
```

○ d.
```
sub normalize_dataset {
 my ($self, $dataset, $minmax) = @_;
 my $min = $minmax->slice_axis(axis => 0, begin => 0, end => 1);
 my $max = $minmax->slice_axis(axis => 0, begin => 1, end => 2);
 $dataset->slice( [0, $dataset->shape->[0] -1], [0, $dataset->shape->[1] -1]) .= ($dataset - $min) / ($max - $min);
}
```

○ e.
```
sub normalize_dataset {
 my ($self, $dataset, $minmax) = @_;
 my $min = $minmax->slice_axis(axis => 1, begin => 0, end => 1)->reshape([-1]);
 my $max = $minmax->slice_axis(axis => 1, begin => 1, end => 2)->reshape([-1]);
 $dataset->slice( [0, $dataset->shape->[0] -1], [0, $dataset->shape->[1] -1]) .= ($dataset - $min) / ($max - $min);

}
```

Respuesta incorrecta.

La respuesta correcta es:
```
sub normalize_dataset{
 my ($self, $dataset, $minmax) = @_;


 my $min = $minmax->slice_axis(axis=>1, begin=>0, end=>1)->T;
 my $max = $minmax->slice_axis(axis=>1, begin=>1, end=>2)->T;
 $dataset->slice( [0, $dataset->shape->[0] -1], [0, $dataset->shape->[1] -1]) .= ($dataset - $min) / ($max - $min);

}
```

Pregunta **3**
Correcta
Puntúa 3,33 sobre 3,33
Marcar pregunta

Considere el siguiente algoritmo dado en la forma escalar:

```
sub column_stdevs{
 my ($self, $dataset, $means) = @_;
 my $stdevs = [0, map {$_} 0 .. $#{$dataset->[0]} -1];
 for my $i (0 .. $#{$dataset->[0]}){
  my $variance = [map {($_->[$i] - $means->[$i]) ** 2} @$dataset];
  $stdevs->[$i] = sum(@$variance);
 }
 $stdevs = [map {sqrt($_ / (scalar(@$dataset) -1))} @$stdevs];
 return $stdevs;
}
```

Elija la opción que correctamente convierte el código escalar a la forma tensorial equivalente, la cual arroja los mismos resultados.

Seleccione una:

○ a.
```
sub column_stdevs {
 my ($self, $dataset, $means) = @_;
 return mx->nd->sqrt(($dataset - $means)->power(2)->mean(axis => 0));
}
```

● b.
```
sub column_stdevs{
 my ($self, $dataset, $means) = @_;
 return mx->nd->sqrt(($dataset - $means)->power(2)->sum(axis=>0) / ($dataset->len - 1));
}
```
✓

- Se calcula `(dataset - means)^2` de forma vectorizada.
- Se suman los cuadrados a lo largo de las filas (`axis => 0`) para obtener la varianza por columna.
- Se divide por `n - 1` (muestra), y luego se toma la raíz cuadrada para obtener la desviación estándar por columna.

○ c.
```
sub column_stdevs {
 my ($self, $dataset, $means) = @_;
 return mx->nd->sqrt($dataset->power(2)->sum(axis => 0) / ($dataset->len - 1));
}
```

○ d.
```
sub column_stdevs {
 my ($self, $dataset, $means) = @_;
 return mx->nd->sqrt(($dataset - $means)->power(2)->sum(axis => 1) / ($dataset->len - 1));
}
```

○ e.
```
sub column_stdevs {
 my ($self, $dataset, $means) = @_;
 return mx->nd->sqrt(($dataset - $means)->sum(axis => 0)->power(2) / ($dataset->len - 1));
}
```

Respuesta correcta

La respuesta correcta es:
```
sub column_stdevs{
 my ($self, $dataset, $means) = @_;
 return mx->nd->sqrt(($dataset - $means)->power(2)->sum(axis=>0) / ($dataset->len - 1));
}
```

Pregunta **4**
Correcta
Puntúa 3,33 sobre 3,33
Marcar pregunta

Considere el siguiente algoritmo dado en la forma escalar:

```
sub standardize_dataset{
 my ($self, $dataset, $means, $stdevs) = @_;
 for my $row (@$dataset){
  for my $i (0 .. $#$row){
   $row->[$i] = ($row->[$i] - $means->[$i]) / $stdevs->[$i];
  }
 }
}
```

Elija la opción que correctamente convierte el código escalar a la forma tensorial equivalente, la cual arroja los mismos resultados.

Seleccione una:

○ a.
```
sub standardize_dataset {
 my ($self, $dataset, $means, $stdevs) = @_;
 $dataset .= ($means - $dataset) / $stdevs;
}
```

○ b.
```
sub standardize_dataset {
 my ($self, $dataset, $means, $stdevs) = @_;
 $dataset->slice([0, $dataset->shape->[0] -1], [0, $dataset->shape->[1] -1]) .=
 ($dataset + $means) / $stdevs;
}
```

○ c.
```
sub standardize_dataset {
 my ($self, $dataset, $means, $stdevs) = @_;
 $dataset->slice([0, $dataset->shape->[0] -1], [0, $dataset->shape->[1] -1]) .=
 ($dataset - $means) * $stdevs;
}
```

● d.
```
sub standardize_dataset{
 my ($self, $dataset, $means, $stdevs) = @_;
 $dataset->slice([0, $dataset->shape->[0] -1], [0, $dataset->shape->[1] -1]) .=
 ($dataset - $means) / $stdevs;
}
```
✓ Correcta porque:

- Utiliza operaciones vectorizadas.
- Modifica el tensor en memoria (`in-place`) mediante slicing.
- Aplica la fórmula estadística correcta.

○ e.
```
sub standardize_dataset {
 my ($self, $dataset, $means, $stdevs) = @_;
 my $standardized = ($dataset - $means) / $stdevs;
 return $standardized;
}
```

Respuesta correcta

La respuesta correcta es:
```
sub standardize_dataset{
 my ($self, $dataset, $means, $stdevs) = @_;
 $dataset->slice([0, $dataset->shape->[0] -1], [0, $dataset->shape->[1] -1]) .=
 ($dataset - $means) / $stdevs;
}
```

Pregunta **5**
Correcta
Puntúa 3,33 sobre 3,33
Marcar pregunta

Considere el siguiente algoritmo dado en la forma escalar:

```
sub train_test_split{
 my ($self, $dataset, %args) = (splice (@_, 0, 2), split=>0.6, @_);
 my $train_size = int($args{split} * @$dataset);
 my @idx       = shuffle (0 .. $#$dataset);
 my @train_idx  = @idx[0 .. $train_size -1];
 my @test_idx   = @idx[$train_size .. $#$dataset];
 my @train      = @$dataset[@train_idx];
 my @test       = @$dataset[@test_idx];
 return \@train, \@test;
}
```

Elija la opción que correctamente convierte el código escalar a la forma tensorial equivalente, la cual arroja los mismos resultados.

Seleccione una:

○ a.
```
sub train_test_split {
 my ($self, $dataset, %args) = (splice(@_, 0, 2), split => 0.6, @_);

 my $train_size = int($args{split} * $dataset->len);
 my $idx       = mx->nd->arange(stop => $dataset->len)->shuffle;
 my $train_idx  = $idx->slice([$train_size, $dataset->len - 1]);
 my $test_idx   = $idx->slice([0, $train_size - 1]);
 my $train      = mx->nd->take($dataset, $train_idx);
 my $test       = mx->nd->take($dataset, $test_idx);

 return $train, $test;
}
```

● b.
```
sub train_test_split{
 my ($self, $dataset, %args) = (splice (@_, 0, 2), split=>0.6, @_);

 my $train_size = int($args{split} * $dataset->len);
 my $idx       = mx->nd->arange(stop => $dataset->len)->shuffle;
 my $train_idx  = $idx->slice([0, $train_size -1]);
 my $test_idx   = $idx->slice([$train_size, $dataset->len -1]);
 my $train      = mx->nd->take($dataset, $train_idx);
 my $test       = mx->nd->take($dataset, $test_idx);

 return $train, $test;
}
```
✓

1. **Uso correcto de arange y shuffle**:
   - `mx->nd->arange(stop => $dataset->len)` genera un tensor con índices `0 .. len-1`.
   - El método `.shuffle` reordena estos índices aleatoriamente, lo cual garantiza una partición aleatoria del dataset.
2. **División precisa y sin solapamientos**:
   - La lista de índices aleatorios se divide en dos partes:
     - Los primeros `$train_size` índices se usan para el conjunto de entrenamiento.
     - El resto se usa para el conjunto de prueba.
   - Esta división se hace **sobre la misma secuencia barajada**, lo que evita duplicados o pérdidas de datos.
3. **Selección de subconjuntos con `nd.take`**:
   - `mx->nd->take($dataset, $train_idx)` y `mx->nd->take($dataset, $test_idx)` permiten seleccionar directamente las filas del dataset, conservando la forma del tensor y sin errores de tipo o dimensión.
4. **Generalización controlada**:
   - El valor por defecto del parámetro `split => 0.6` se puede sobrescribir fácilmente.
   - La lógica se adapta a datasets de cualquier tamaño sin romperse.

○ c.
```
sub train_test_split {
 my ($self, $dataset, %args) = (splice(@_, 0, 2), split => 0.6, @_);

 my $train_size = int($args{split} * $dataset->len);
 my $idx       = mx->nd->arange(stop => $dataset->len)->shuffle;
 my $train_idx  = $idx->slice([0, $train_size - 1]);
 my $test_idx   = mx->nd->arange(stop => $dataset->len)->shuffle->slice([$train_size, $dataset->len - 1]);
 my $train      = mx->nd->take($dataset, $train_idx);
 my $test       = mx->nd->take($dataset, $test_idx);

 return $train, $test;
}
```

○ d.
```
sub train_test_split {
 my ($self, $dataset, %args) = (splice(@_, 0, 2), split => 0.6, @_);

 my $train_size = int($args{split} * $dataset->shape->[0]);
 my $idx       = mx->nd->arange(stop => $dataset->len);
 my $train_idx  = $idx->slice([0, $train_size - 1]);
 my $test_idx   = $idx->slice([$train_size, $dataset->len - 1]);
 my $train      = mx->nd->take($dataset, $train_idx);
 my $test       = mx->nd->take($dataset, $test_idx);

 return $train, $test;
}
```

○ e.
```
sub train_test_split {
 my ($self, $dataset, %args) = (splice(@_, 0, 2), split => 0.6, @_);

 my $train_size = int($args{split} * $dataset->len);
 my $train      = $dataset->slice([0, $train_size - 1], 'X');
 my $test       = $dataset->slice([$train_size, $dataset->len - 1], 'X');

 return $train, $test;
}
```

Respuesta correcta

La respuesta correcta es: sub train_test_split{