Which is best?

Why not choose the method with the best fit to the data?
What do we really want?

Why not choose the method with the best fit to the data?

“How well are you going to predict future data drawn from the same distribution?”
The test set method

1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set
The test set method

1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set

(Linear regression example)
The test set method

1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set
4. Estimate your future performance with the test set

(Linear regression example)
Mean Squared Error = 2.4
The test set method

1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set
4. Estimate your future performance with the test set

(Quadratic regression example)
Mean Squared Error = 0.9
The test set method

1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set
4. Estimate your future performance with the test set

(Join the dots example)

Mean Squared Error = 2.2
The test set method

Good news:

• Very very simple
• Can then simply choose the method with the best test-set score

Bad news:

• What’s the downside?
The test set method

Good news:
- Very very simple
- Can then simply choose the method with the best test-set score

Bad news:
- Wastes data: we get an estimate of the best method to apply to 30% less data
- If we don’t have much data, our test-set might just be lucky or unlucky

We say the “test-set estimator of performance has high variance”
LOOCV (Leave-one-out Cross Validation)

For $k=1$ to $R$

1. Let $(x_k, y_k)$ be the $k^{th}$ record
LOOCV (Leave-one-out Cross Validation)

For $k=1$ to $R$

1. Let $(x_k, y_k)$ be the $k^{th}$ record

2. Temporarily remove $(x_k, y_k)$ from the dataset
LOOCV (Leave-one-out Cross Validation)

For $k=1$ to $R$

1. Let $(x_k, y_k)$ be the $k^{th}$ record
2. Temporarily remove $(x_k, y_k)$ from the dataset
3. Train on the remaining $R-1$ datapoints
LOOCV (Leave-one-out Cross Validation)

For \( k = 1 \) to \( R \)

1. Let \((x_k, y_k)\) be the \( k^{th} \) record
2. Temporarily remove \((x_k, y_k)\) from the dataset
3. Train on the remaining \( R-1 \) datapoints
4. Note your error \((x_k, y_k)\)
LOOCV (Leave-one-out Cross Validation)

For $k=1$ to $R$

1. Let $(x_k, y_k)$ be the $k^{th}$ record

2. Temporarily remove $(x_k, y_k)$ from the dataset

3. Train on the remaining $R-1$ datapoints

4. Note your error $(x_k, y_k)$

When you’ve done all points, report the mean error.
LOOCV (Leave-one-out Cross Validation)

For k = 1 to R

1. Let \((x_k, y_k)\) be the \(k^{th}\) record

2. Temporarily remove \((x_k, y_k)\) from the dataset

3. Train on the remaining \(R-1\) datapoints

4. Note your error \((x_k, y_k)\)

When you’ve done all points, report the mean error.

\[ \text{MSE}_{\text{LOOCV}} = 2.12 \]
LOOCV for Quadratic Regression

For k=1 to R
1. Let \((x_k, y_k)\) be the \(k^{th}\) record
2. Temporarily remove \((x_k, y_k)\) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error \((x_k, y_k)\)

When you've done all points, report the mean error.

\[ MSE_{LOOCV} = 0.962 \]
For $k=1$ to $R$

1. Let $(x_k, y_k)$ be the $k^{th}$ record

2. Temporarily remove $(x_k, y_k)$ from the dataset

3. Train on the remaining $R-1$ datapoints

4. Note your error $(x_k, y_k)$

When you've done all points, report the mean error.

$$MSE_{LOOCV} = 3.33$$
Which kind of Cross Validation?

<table>
<thead>
<tr>
<th></th>
<th>Downside</th>
<th>Upside</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test-set</td>
<td>Variance: unreliable estimate of future performance</td>
<td>Cheap</td>
</tr>
<tr>
<td>Leave-one-out</td>
<td>Expensive. Has some weird behavior</td>
<td>Doesn't waste data</td>
</tr>
</tbody>
</table>

..can we get the best of both worlds?
k-fold Cross Validation

Randomly break the dataset into \( k \) partitions (in our example we’ll have \( k=3 \) partitions colored Red Green and Blue)
k-fold Cross Validation

Randomly break the dataset into k partitions (in our example we’ll have k=3 partitions colored Red Green and Blue)

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.
Randomly break the dataset into k partitions (in our example we’ll have k=3 partitions colored Red Green and Blue)

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.
Randomly break the dataset into \( k \) partitions (in our example we’ll have \( k=3 \) partitions colored Red Green and Blue):

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.
Randomly break the dataset into $k$ partitions (in our example we’ll have $k=3$ partitions colored Red Green and Blue)

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Then report the mean error
k-fold Cross Validation

Randomly break the dataset into $k$ partitions (in our example we’ll have $k=3$ partitions colored Red Green and Blue)

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Then report the mean error

$\text{MSE}_{3\text{FOLD}}=1.11$
k-fold Cross Validation

Randomly break the dataset into $k$ partitions (in our example we’ll have $k=3$ partitions colored Red Green and Blue)

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Then report the mean error $MSE_{3FOLD} = 2.93$
## Which kind of Cross Validation?

<table>
<thead>
<tr>
<th></th>
<th>Downside</th>
<th>Upside</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test-set</strong></td>
<td>Variance: unreliable estimate of future performance</td>
<td>Cheap</td>
</tr>
<tr>
<td><strong>Leave-one-out</strong></td>
<td>Expensive. Has some weird behavior</td>
<td>Doesn’t waste data</td>
</tr>
<tr>
<td><strong>10-fold</strong></td>
<td>Wastes 10% of the data. 10 times more expensive than test set</td>
<td>Only wastes 10%. Only 10 times more expensive instead of $R$ times.</td>
</tr>
<tr>
<td><strong>3-fold</strong></td>
<td>Wastier than 10-fold. Expensivier than test set</td>
<td>Slightly better than test-set</td>
</tr>
<tr>
<td><strong>R-fold</strong></td>
<td>Identical to Leave-one-out</td>
<td></td>
</tr>
</tbody>
</table>