LIME

What it computes, and which half of that you can trust

Seminar

William Bendinelli

Interpretable Machine Learning · Guest Lecture

ACT 1

What it computes

The four words, the equation, and the six steps that implement it.

ACT 2

Where it falls short

Two difficulties of the method, measured here — not just cited.

ACT 3

How to read it

Which half of the output to quote, and four cheap checks.

ONE DATASET THROUGHOUT

- Digitised images of cells drawn from **569 breast tumours** by fine-needle aspiration
- What the model has to predict: **is this tumour malignant or benign?**
- **30 measurements** per patient — 10 features of the cell nuclei, each reported three ways: mean, standard error, and "worst"
- That 10 × 3 structure — and the geometry tying radius, perimeter and area together — is **why the measurements are so correlated**

ACT 1

What it computes

What the name promises.

L OCAL	It only has to be right near this one patient. Anywhere else it may be wrong.	local fidelity
I NTERPRETABLE	The simple model must be one a person can read: a line, or a short tree.	surrogate model
M ODEL-AGNOSTIC	You never look inside the model. You only send it data and read the answers.	black-box access
E XPLANATIONS	You read the simple model, not the real one.	feature weights

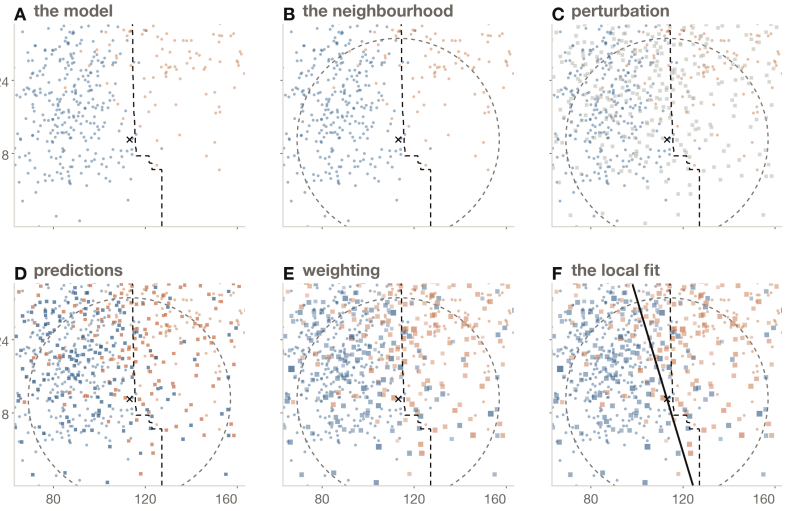
What the equation asks for.

Find the simplest model that behaves like the real one, near this patient.

$$\xi(x) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

find a model	$\arg\min_{g \in G}$	search inside G — the models a person can read
behaves like the real one	$\mathcal{L}(f, g, \pi_x)$	how far the simple model sits from the real one
near this patient	π_x	gives more weight to points close by
the simplest	$\Omega(g)$	penalises complication

What comes out is $\xi(x)$ — the explanation. Ribeiro et al. (2016)

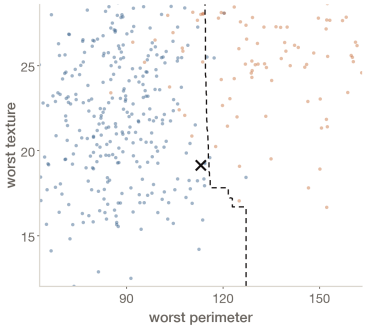


patient	**#67**
P(benign)	**0.581**
R^2 of the local fit	**0.36**

every panel: worst perimeter (x) and worst texture (y), ±1.5σ around the patient

- real patient · benign
- real patient · malignant
- × patient #67
- neighbour, no label yet
- neighbour → benign
- neighbour → malignant
- the model's boundary
- kernel ring, w = 0.95
- the local fit

THE PICTURE



● benign ● malignant × patient #67
Dashed line: the forest's own boundary in this slice.
Patient #67 sits on it, so the geometry stays visible.

THE CODE

```
X, y = load_breast_cancer(
    return_X_y=True)      # 569 x 30

(X_train, X_test,
 y_train, y_test) = train_test_split(
    X, y, test_size=0.25,   # 426/143
    stratify=y, random_state=42)

model = RandomForestClassifier(
    n_estimators=300,
    min_samples_leaf=3,
    random_state=42,
).fit(X_train, y_train)

# f is all LIME ever gets: rows in,
# probabilities out. class 1 = benign
f = model.predict_proba
x = X_test[67]          # the patient
f([x])[0, 1]            # -> 0.5812
```

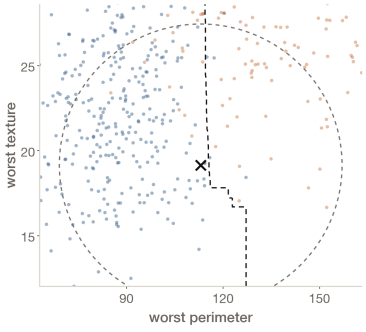
THE IDEA

Explain one prediction, not the model.

- A 300-tree forest is inspectable in principle and unreadable in practice — so we treat it as a black box and only ask it questions.
- Patient #67 sits at 0.58 — near the boundary, so there is geometry to see.
- The dashed line is that boundary in this plane, the other 28 frozen.

SO FAR — a model, and one row.

THE PICTURE



● benign ● malignant × patient #67
Dashed ring: the kernel's 0.95-weight contour, at 1.32σ. Rows outside it still count, only less.

THE CODE

```
# every feature on its own scale —
# distances live there, never in
# the raw units
mu    = X_train.mean(axis=0)
sigma = X_train.std(axis=0)
x_scaled = (x - mu) / sigma

# ONE width for all 30, a scalar
# lime_tabular.py:243
nu = 0.75 * np.sqrt(30)   # 4.108

# what that width is worth, in sigma
r95  = nu * np.sqrt(-2*np.log(0.95))
half = nu * np.sqrt(2*np.log(2))
print(r95, half)   # -> 1.32, 4.84
```

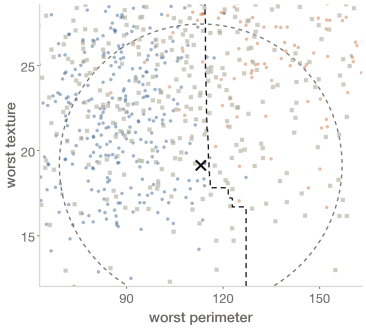
THE IDEA

Locality must be defined before it means anything.

- We choose the distance and the width nu. The data chooses neither.
- nu says how fast a row stops counting. Here nu = 4.11 — the package's own default.
- Only the 0.95 contour fits this frame. Act 2 shows where the other one sits.

SO FAR — + a definition of "near".

THE PICTURE



● benign ● malignant × patient #67
■ synthetic neighbour
Squares: 500 of the 5,000 probe rows; 275 land inside the frame. All 30 features are resampled.

THE CODE

```
rng = np.random.RandomState(42)

# sample_around_instance=False is the
# default: the cloud is centred on the
# DATASET, not on the patient. We also
# set discretize_continuous=False —
# the default explains quartile BINS.
Z = mu + sigma * rng.normal(
        0, 1, size=(5000, 30))
Z[0] = x    # lime_tabular.py:537

Zs = (Z - mu) / sigma

# the cloud's centre is the origin.
# the patient is 2.95 sigma away:
print(np.linalg.norm(Zs.mean(axis=0)),
      np.linalg.norm(x_scaled))
# -> 0.08   2.95
```

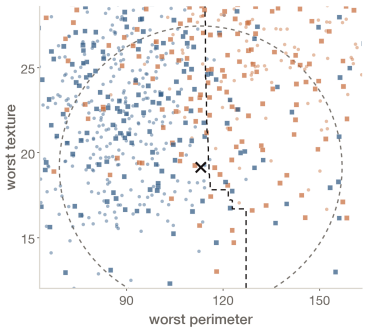
THE IDEA

If you cannot look inside, build data to probe with.

- These are not perturbations of the patient. They are 5,000 rows drawn from a Gaussian fitted to the training data — each feature sampled on its own.
- Each of the 30 features is drawn on its own — the correlations are gone.
- Patient #67 sits 2.95σ from the centre of that cloud.

SO FAR — + 5,000 probe rows.

THE PICTURE



● benign ● malignant × patient #67
■ pred. benign ■ pred. malignant
Square colour is the model's own verdict, not a diagnosis: these rows were never biopsied.

THE CODE

```
# the only time LIME itself calls f
# — 5,000 rows of 30 numbers, one
# call. these are the labels the
# explanation is fitted on.
fz = f(Z)[:, 1]

# raw Z, not Zs: the forest was
# trained in raw units
fz.mean()                  # -> 0.4837

# so the average probe row is a coin
# flip. hers is 0.5812. this cloud
# is not a neighbourhood.
```

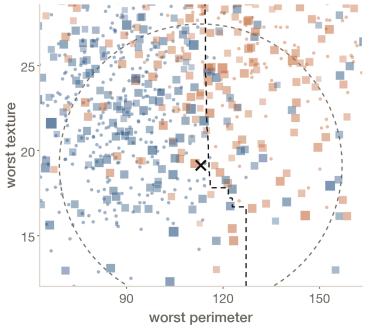
THE IDEA

The black box becomes its own teacher.

- The one moment LIME queries the model. 5,000 rows in, 5,000 labels out.
- Only 68% obey the drawn boundary, against a 53% baseline.
- They cannot be expected to: each row carries random values on the other 28 features.

SO FAR — + the model's answer on each.

THE PICTURE



● benign ● malignant × patient #67
■ pred. benign ■ pred. malignant
Size and opacity encode π unnormalised. The × marks π = 1 (not drawn to scale); the largest square is 0.69.

THE CODE

```
# distance over all 30 STANDARDIZED
# features, not the two we plot
d = np.linalg.norm(
      Zs - x_scaled, axis=1)

# lime_tabular.py:248 — the sqrt
# makes it exp(-d^2 / 2 nu^2)
# w is pi_x(z) from the objective
w = np.sqrt(
      np.exp(-d**2 / nu**2))

print(w.min(), w.max())
# -> 0.083  1.000. the 1.000 is the
#    patient's own row: Z[0] = x

# effective sample size, in rows:
# weighting costs the equivalent
# of 355 of the 5,000 rows
print(w.sum()**2 / (w**2).sum())
# -> 4645
```

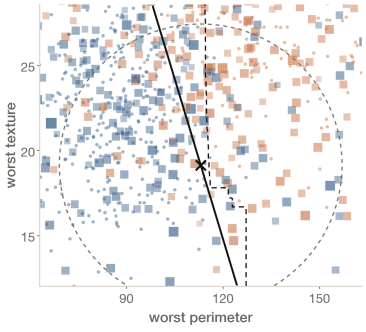
THE IDEA

This is the only step that makes the fit local.

- Every row now carries a weight set by its distance.
- Everything before this was global.
- This is the step meant to make it local — and the first of the two steps Act 2 takes apart.

SO FAR — + a weight per row.

THE PICTURE



● benign ● malignant × patient #67
■ pred. benign ■ pred. malignant
Solid line: the fit itself, seen in this plane and drawn through the patient rather than at P = 0.5.

THE CODE

```
# stage 1 — rank 30, keep 8. LIME
# defaults to 10. alpha=0.01 is
# almost OLS: it only sorts.
aux  = Ridge(alpha=0.01).fit(
    Zs, fz, sample_weight=w)

# coef x HER standardized value —
# raw units pick a different eight.
top8 = np.argsort(-np.abs(
    aux.coef_ * x_scaled))[:8]

# stage 2 — the explanation itself
g = Ridge(alpha=1).fit(
    Zs[:, top8], fz,
    sample_weight=w)
r2 = g.score(Zs[:, top8], fz,
             sample_weight=w)
gx = g.predict([x_scaled[top8]])[0]
print(r2, gx)   # 0.3572  0.4968
```

THE IDEA

The fitted model is the explanation.

- Two weighted stages: rank the 30, keep 8, then fit only those 8.
- The eight coefficients are the explanation. The forest's own numbers never appear.
- And the 8 is our choice — LIME defaults to 10.

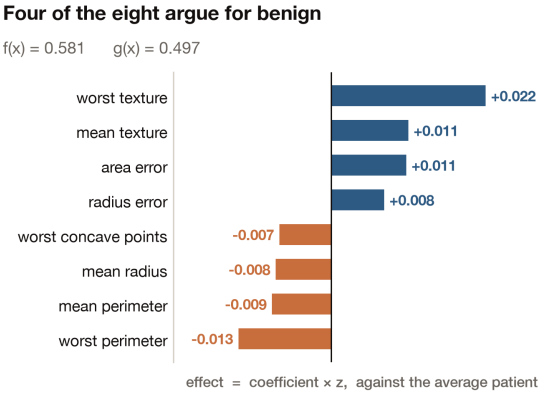
DONE — 8 coefficients, and $R^2 = 0.36$.

What the explanation says about #67.

01 Each bar is an *effect*: the coefficient times how far #67 sits from the average, in standard deviations. Molnar's chapter plots the same object

02 Four of the eight push toward malignant, four toward **benign**

03 The library prints the bare coefficients instead — all eight of those are negative. Multiplying by where the patient sits is what splits them



ACT 2

Where it falls short

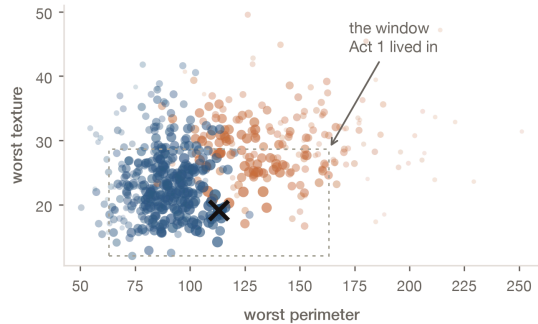
The “local” fit is barely local.

01 Garreau & von Luxburg (2020b) proved it: at the default width nu = 0.75√p the kernel “becomes redundant — it is equivalent to give weight 1 to every perturbed sample”

02 We measured: delete it entirely and g(x) moves 0.002 — median of six patients, 0.013 at worst — and the same eight features survive. Its half-weight contour sits at 4.84σ, where 42% of all 569 patients count as neighbours

03 And the book agrees it is unsolved: “we don’t have a good way to find the best kernel or width”

Dot size is each patient's actual kernel weight



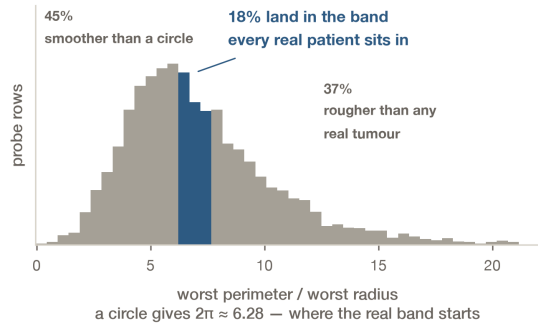
It explains data that cannot exist.

01 The book: the samples come “from the training data’s mass center”, not from around the patient — “which is problematic”

02 We measured it two ways. 76% of the probe rows carry a negative measurement, and 82% have a perimeter-to-radius ratio no real patient has

03 And such rows are easy to spot. Slack et al. (2020) used exactly that to hide a biased classifier: it keeps discriminating, and LIME reports it as innocuous

82% of the probe rows have a shape no real patient has



ACT 3

How to read it

01 Try more than one kernel width

Molnar's own advice: "try different kernel settings and see for yourself if the explanations make sense". Here a 6× sweep of the width — nu from 2.7 to 16.4, the default is 4.1 — changes nothing. That is the check passing, not the check being unnecessary.

02 R^2: the book, or the data?

Molnar lists fidelity as a strength — R^2 "gives us a good idea of how reliable the interpretable model is". Here it swings 30% on the seed alone, and going from 8 features to 30 doubles it while g(x) moves 0.004. Which do you keep? Open for discussion.

03 Quote the direction, never the level

Check g(x) against f(x): one line of code, and the only number here that needs no reference. Here g(x) = 0.497 where f(x) = 0.581 — the level is the half you cannot quote. Median gap 0.222 across 143 patients; 6 land on the opposite side of 0.5.

04 Re-run before you believe a ranking

Molnar's instability limitation, measured: seven of the eight features held across 20 seeds and only the eighth slot rotated between three — but you cannot know that without re-running, and a single run never says which slot moved.

CLOSING

"... the method is still in the development phase and many problems need to be solved before it can be safely applied."

Christoph Molnar

Thanks!

github.com/wbendinelli/interpretable-ml-lectures

REFERENCES

Ribeiro et al. (2016). "Why Should I Trust You?" Explaining the Predictions of Any Classifier. *KDD '16*

Garreau & von Luxburg (2020a). Explaining the Explainer: A First Theoretical Analysis of LIME. *AISTATS*

Garreau & von Luxburg (2020b). Looking Deeper into Tabular LIME. arXiv:2008.11092

Slack et al. (2020). Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods. *AIES '20*

Molnar, C. (2025). *Interpretable Machine Learning*, 3rd ed., ch. 14. christophm.github.io/interpretable-ml-book

Alvarez-Melis & Jaakkola (2018). On the Robustness of Interpretability Methods. arXiv:1806.08049

Street et al. (1993). Nuclear feature extraction for breast tumor diagnosis. *IS&T/SPIE 1905* — the dataset used throughout

William Bendinelli