Lecture 35: Interpretation of Prediction Models

CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader and Chris Tanner

Outline

Variable Importance

Permutation Importance

Interpretation through Predictions

Adding Uncertainty

LIME

Variable Importance

Variable Importance for Tree-Based Models

How does sklearn determine **variable importance** (feature_importance) from a tree-based model?

- It determines the improvement in the loss function every time a predictor is involved in a split.

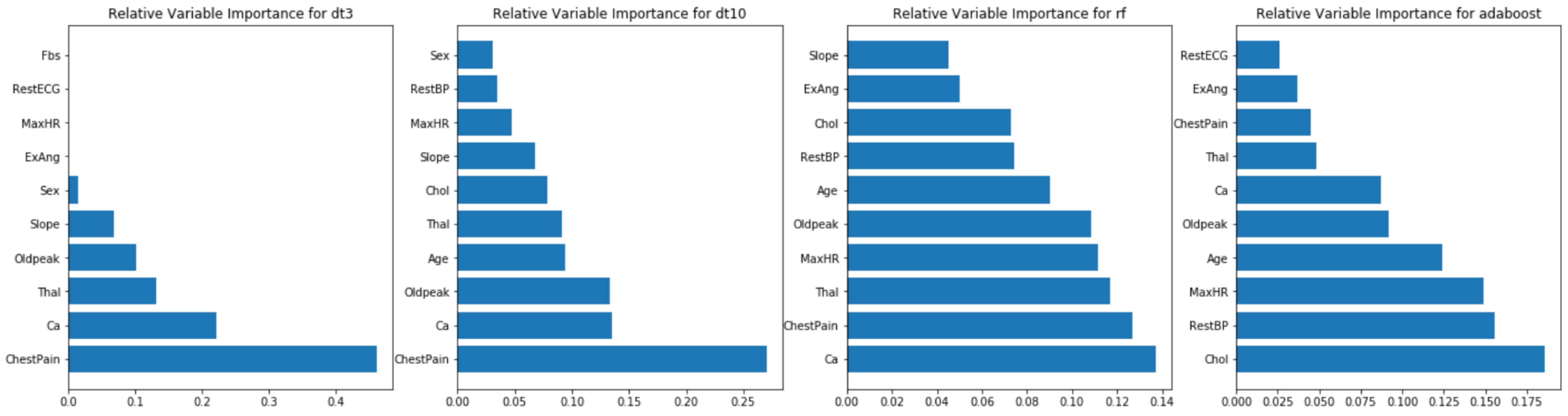
- More specifically, it calculate the total amount that the SSE (for regression) or Gini index (for classification) is improved (decreased) due to splits over a given predictor (averaged over all B trees if a bagged/random forest method).

How should variable importance compare across the various different tree models we've considered (trees, random forests/bagging, and boosting)?

A picture is worth a thousand words…

Variable Importance for trees, bags, and boosts

Below are the variable importance plots for the top 10 predictors for each of a (i) decision tree with maxdepth=3, (ii) decision tree with maxdepth=10, (iii) a random forest, and (iv) an adaboost classifier.



Compare them? Are the differences surprising?

Feature Importance in a Neural Network

How can one measure feature importance in a Neural Network?

Its not so easy ☹. What could potentially be done?

We can calculate the predictions from a neural net, and then fit a decision tree model to the predicted values.

What can go wrong with this approach?

Other Variable Importance Measures

What other approaches can be taken to measure variable importance?

Alternative:

- Record the prediction accuracy on the *oob* samples for each tree.
- Randomly permute the data for column j in the *oob* samples, then record the accuracy again.
- The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of variable j in the random forest.

Permutation Importance

Permutation Variable Importance

This idea of re-permuting a variable and refitting a model to see how much more poorly it performs is called **permutation feature importance.**

It is sometimes preferred to the standard feature importance, why?

When two features are correlated and one of the features is permuted, the model will still have access to the feature through its correlated feature.

What is the one glaring disadvantage to this?

Computational time, and things may not ‘add up’.

Permutation Variable Importance in sklearn

To perform permutation variable importance in sklearn (or keras), one can use the ELI5 library:

https://eli5.readthedocs.io/en/latest/index.html

This is easy to do with sklearn models, but not easy to do with tensorflow (but it can be done).

An example is worth a thousand words:

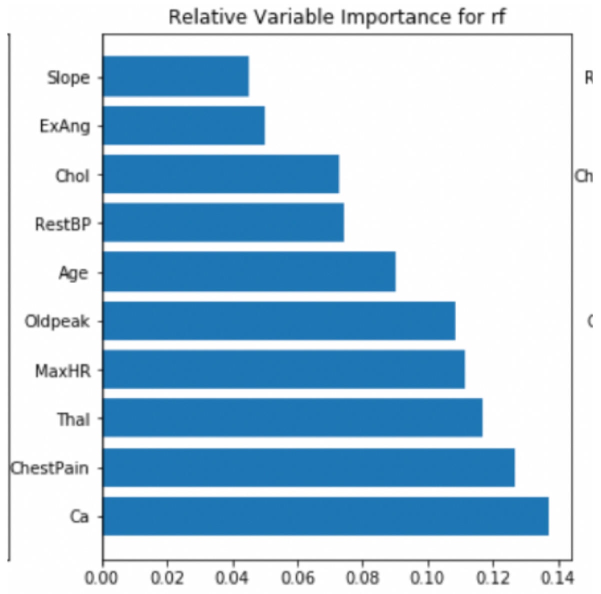
Permutation Variable Importance Example

```python
#permutation importance
from eli5.sklearn import PermutationImportance
from eli5.permutation_importance import get_score_importances


perm = PermutationImportance(randomforest).fit(X_test, y_test)
#eli5.show_weights(perm,feature_names=X.columns)
print(X.columns)
eli5.explain_weights(perm)
```

```
Index(['Age', 'Sex', 'ChestPain', 'RestBP', 'Chol', 'Fbs', 'RestECG', 'MaxHR',
       'ExAng', 'Oldpeak', 'Slope', 'Ca', 'Thal'],
      dtype='object')
```

Weight	Feature
0.1082 ± 0.0161	x12
0.0689 ± 0.0435	x7
0.0393 ± 0.0675	x11
0.0361 ± 0.0636	x2
0.0328 ± 0.0359	x8
0.0295 ± 0.0245	x9
0.0295 ± 0.0131	x1
0.0197 ± 0.0245	x3
0.0164 ± 0.0688	x10
0.0131 ± 0.0131	x6
0.0098 ± 0.0161	x4
0 ± 0.0000	x5
-0.0098 ± 0.0334	x0



The problem with Variable Importance

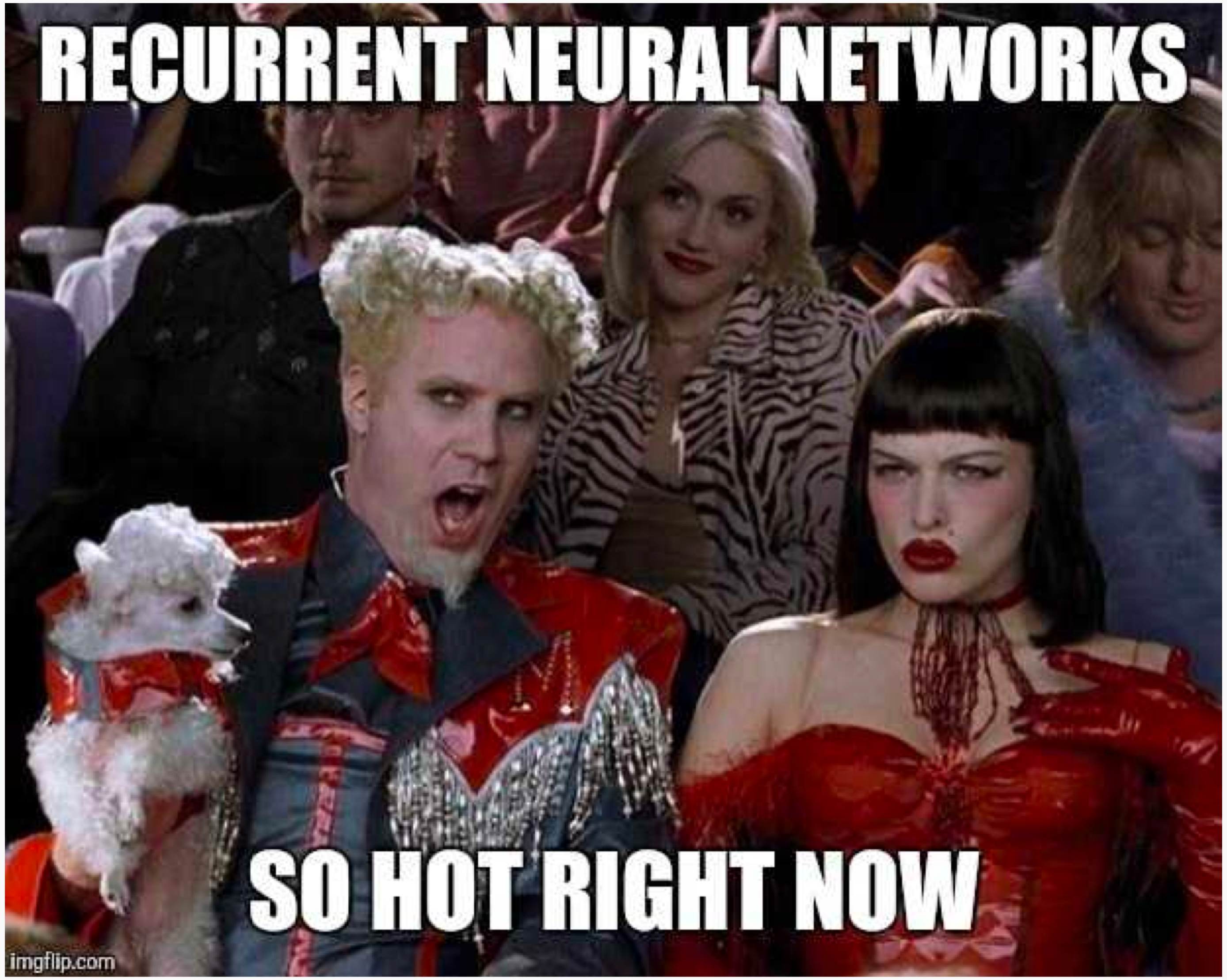
Variable Importance is great! It tells you what features are important in shaping the model.

But what is missing?

- It does not give any measure for how the predictors are related to the response (positive, negative, quasi-linear, curved, interactions, etc.).
- This is where the parametric model wins out! Inference and interpretations are much easier and the whole point in these models.

What can we do to measure these relationships in a machine learning or non-parametric model? Think: what did we do with *k*-NN?

What needs to be done algorithmically to put this in practice?


RECURRENT NEURAL NETWORKS
SO HOT RIGHT NOW
imgflip.com

Interpretation through Predictions

Parametric vs. Nonparametric models

In a machine learning model (like ensemble methods and neural networks), the association between predictors and the response are not measured directly as these models are 'black box' models:

Inputs (X, predictors). → black box (NN, sklearn, etc.) → Outputs (Y, response)

What if we care about how the predictors relate to the response? This is where we need to figure out what the black box is doing to transform the inputs into the outputs.

Simplest Approach: observed $\hat{Y}$ vs. X_j

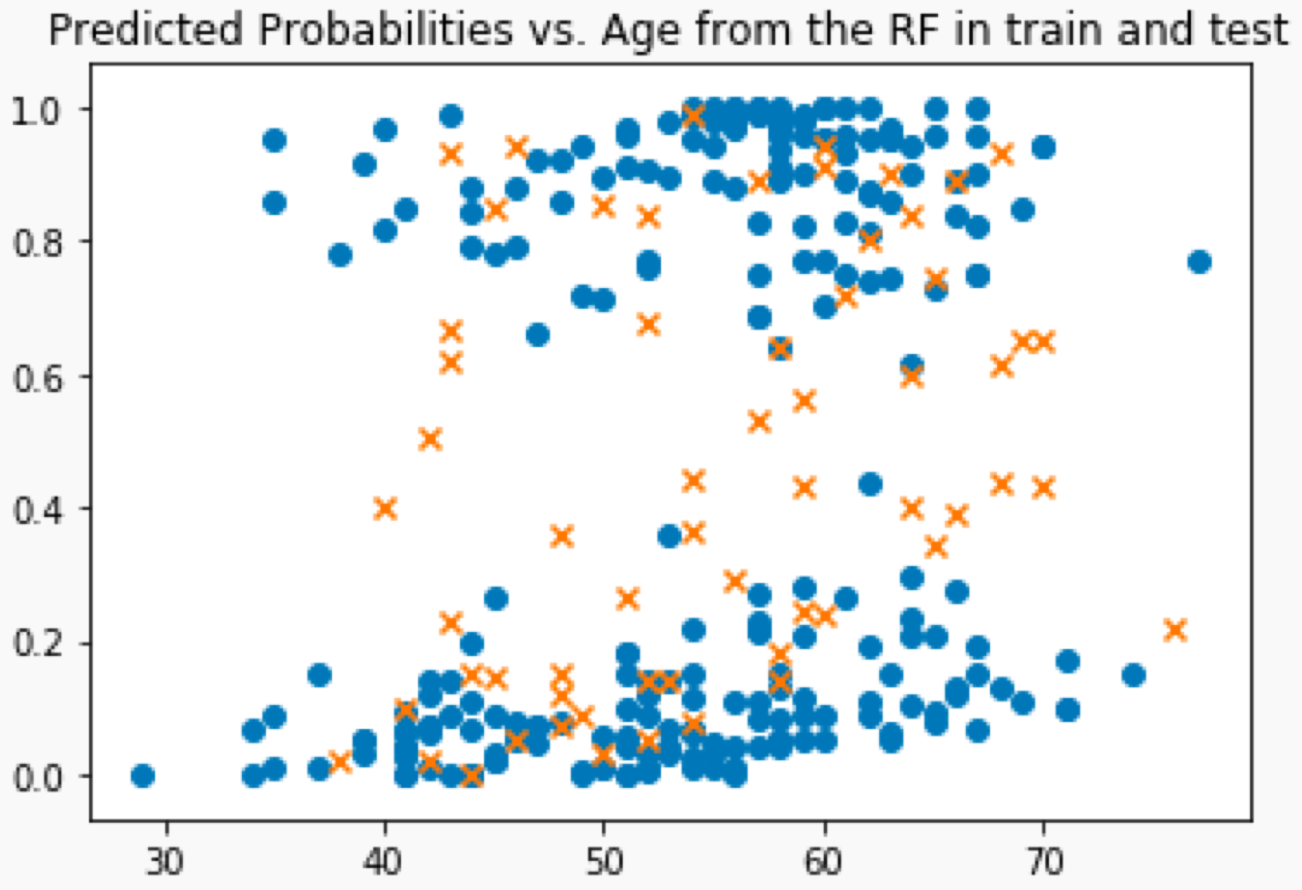
Use predict (or better yet, predict_proba) to plot the observed predicted values vs. the observed values for X_j.

What is a problem with this approach how can we fix it?

The fix is not so easy. We cannot just fit a logistic regression model so easily to the predicted probabilities. Why not?

An example is worth a thousand words…

Example: observed $\hat{Y}$ vs. X_j



Unboxing the black box

Inputs (*X*, predictors). → black box (NN, sklearn, etc.) → Outputs (*Y*, response)

This gives us our approach: vary the inputs (the predictors) and see what happens to the response.

If we care about the 'marginal' or 'conditional' effect of how a specific *X* relates to *Y*, then we should vary only one predictor at a time.

How should we handle the other predictors? That is to say, what value should we keep them at?

Unboxing the black box (cont.)

There are two general approaches to interpreting the machine learning model through predictions:

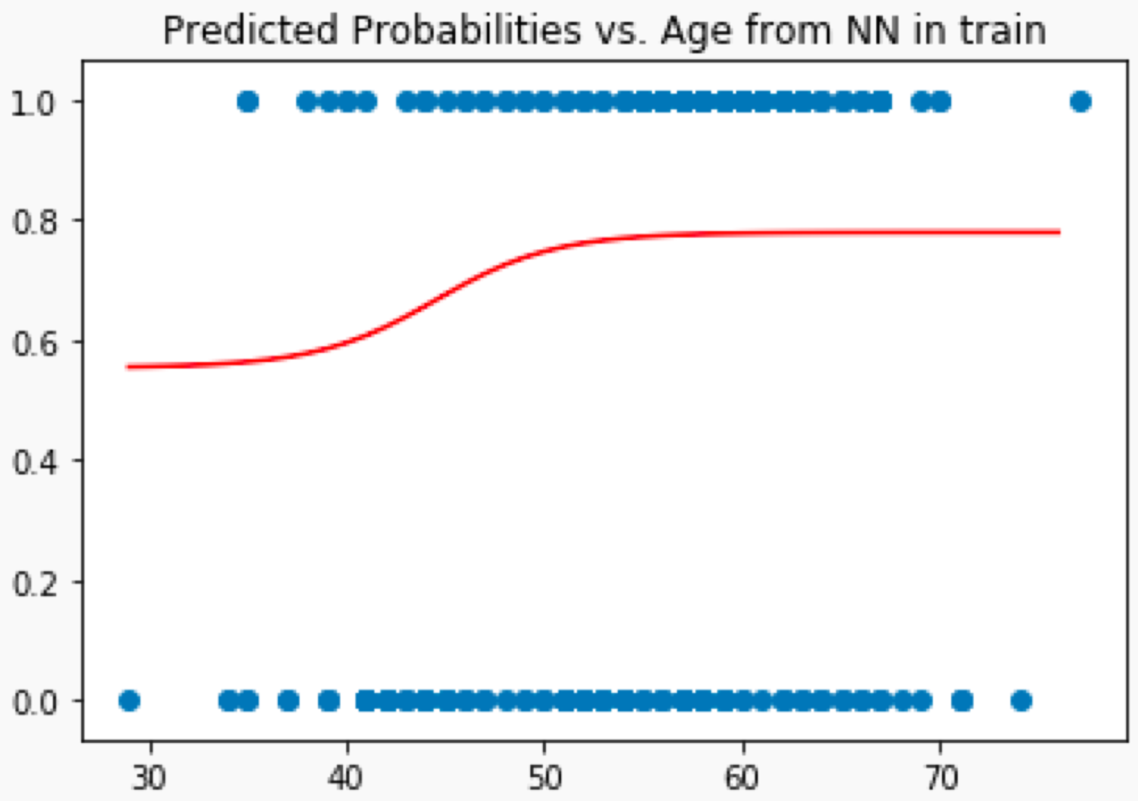
The approach is just like in multiple regression: what is the marginal effect of a unit change in X_j holding all the other predictors constant.

So at what values should we hold the other predictors?

There are two general approaches **holding the other predictors constant**:

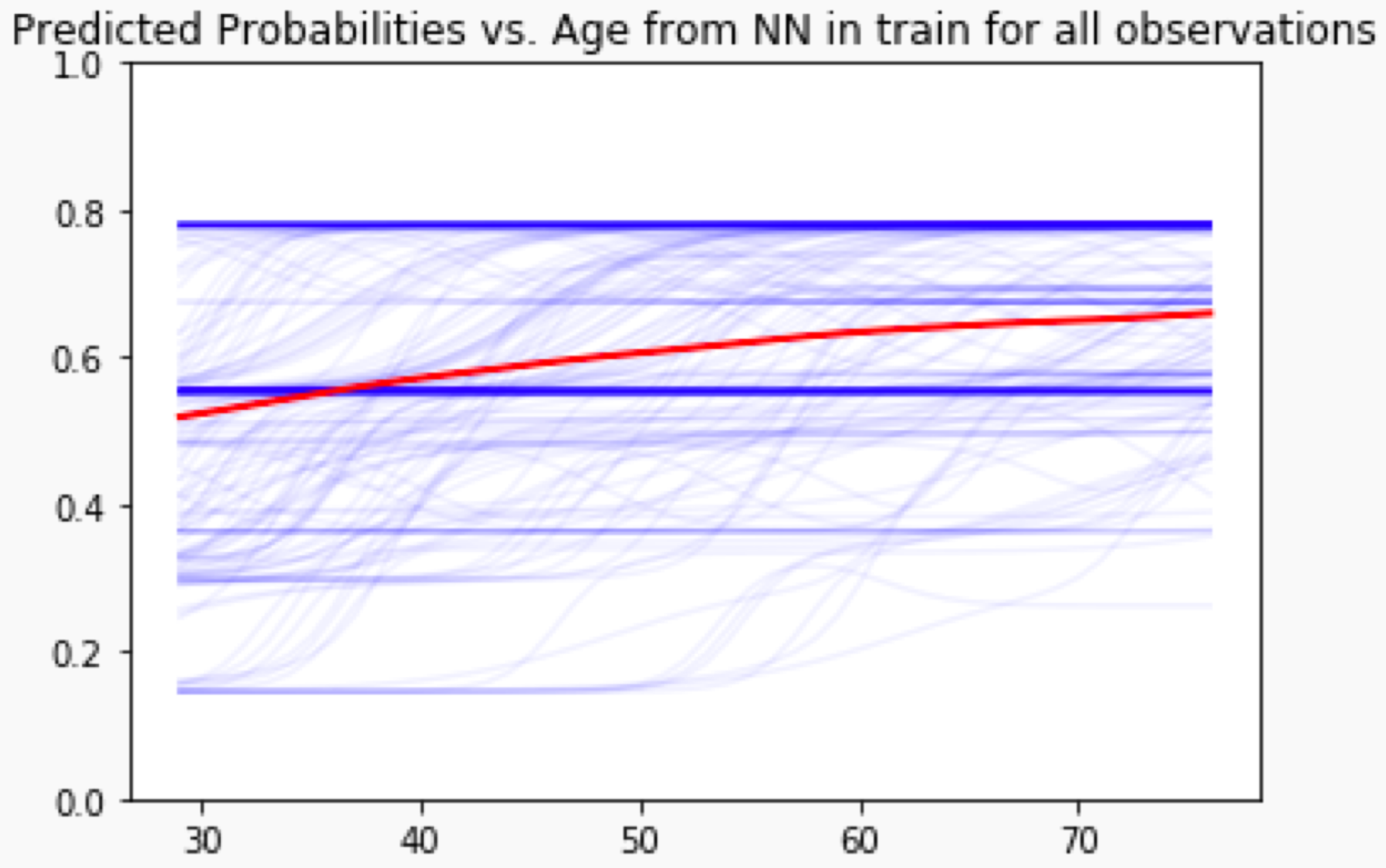
1. Predict $\hat{Y}$ at the **mean (or most common) value** for each of the other predictors, vary only the predictor you care about, X_j, and plot the predictions $\hat{Y}$ vs. X_j.
2. Predict $\hat{Y}$ at the **observed values of** for all the other predictors, vary only the predictor you care about, X_j, and plot the predictions $\hat{Y}$ vs. X_j. Essentially this means creating a new data frame for each observation, and imputing all reasonable values of X_j in.

Predicted probabilities vs. Age at the means



Interpret this plot. What does this say for how Cardiac Arrest is associated with Age?

Predicted probabilities vs. Age for all observations



Interpret this plot. What does this say for how Cardiac Arrest is associated with Age?

What if we want the joint relationship with two predictors?

How can we look at how the response relates to two predictors at once (to directly interpret interactions)?

This is a bit trickier to do. Why?

We can go back to our friend: the classification boundary!

Should I use PCA components?

What if we want an overall picture of how the response relates to the predictors?

Well, this is where PCA components sometimes come in to play.

But this is no Bueno for interpretability. Why?

Adding Uncertainty

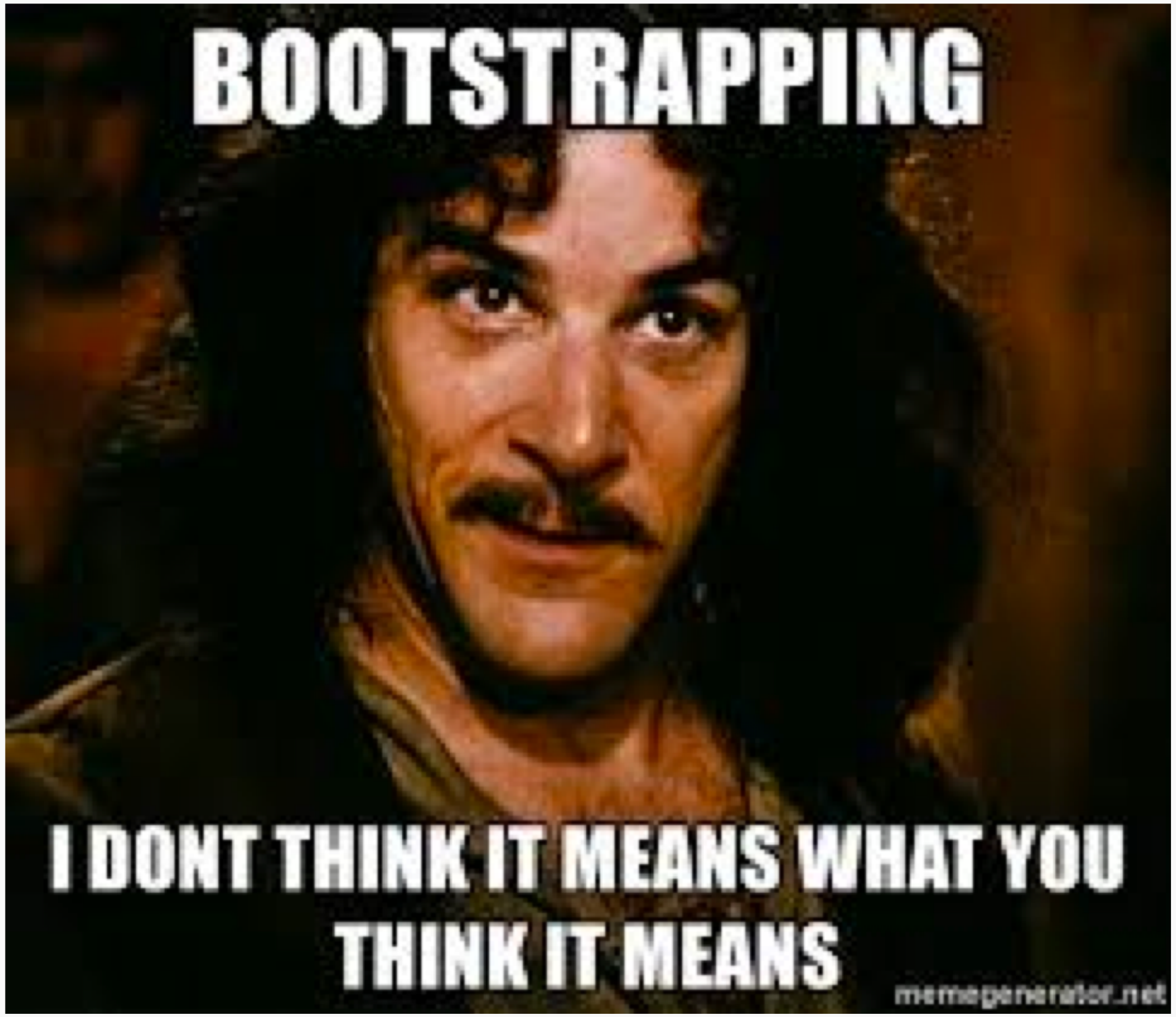
Adding Uncertainty

Not only should we plot predictions, but we can (should) also include the uncertainty of these predictions.

For example, there may be a big difference between.

How can we add uncertainty measures to these predictions? How do we do it for linear/logistic regression?

Bootstrapping!


BOOTSTRAPPING
I DONT THINK IT MEANS WHAT YOU THINK IT MEANS
memegenerator.net

Bootstrapping Basics, review

How do we perform a bootstrap? What are the key steps?

The bootstrap algorithm is an approach to mimic the uncertainty of taking the observed sample from a population. So we:

1. Sample the same number of observations as was observed (in train).
2. Sample with replacement (so we get some randomness).

Why does this work?

Bootstrapping Basics, review

When is a bootstrapping approach used?

1. When a probabilistic closed form solution is not known or not easy to get at.
2. When the assumptions for a parametric model break down.
3. When there is no ‘formula’ at all to add the uncertainty into a calculation.
4. Or we are just being lazy.

Bootstrapping Basics, review

How do we use the results of a bootstrap technique?

It provides us a re-created **estimate** every time we bootstrap.

This estimate could be a mean, a beta, a plot of average predictions, etc.

But they will almost always be for a summary, not a single observation, and thus the sample size is already taken care of (the *n* in the formula). Aka, do not divide by sqrt(n) again!

Bootstrapping for Predictions

So we can refit a model on a bootstrap sample of data, and look at the predicted probabilities each time.

Then we can take the top 97.5th percentile and bottom 2.5th percentile to build a 95% confidence interval for the predictions!

But how does this work for random forests and bagging models? Yikes!

LIME

LIME

One package that can be very useful to help interpret our machine learning models is **LIME**:

Local

Interpretable

Model-agnostic

Explanations

What do each of those contributions represent?

LIME (cont.)

Interpretable: The explanation must be easy to understand by humans (but may depend on the target audience).

Local fidelity: The explanation should be able to explain how the model behaves for individual predictions.

Model-agnostic: The method should be able to explain any model.

Explanations: self-explanatory ☺. This is the whole goal.

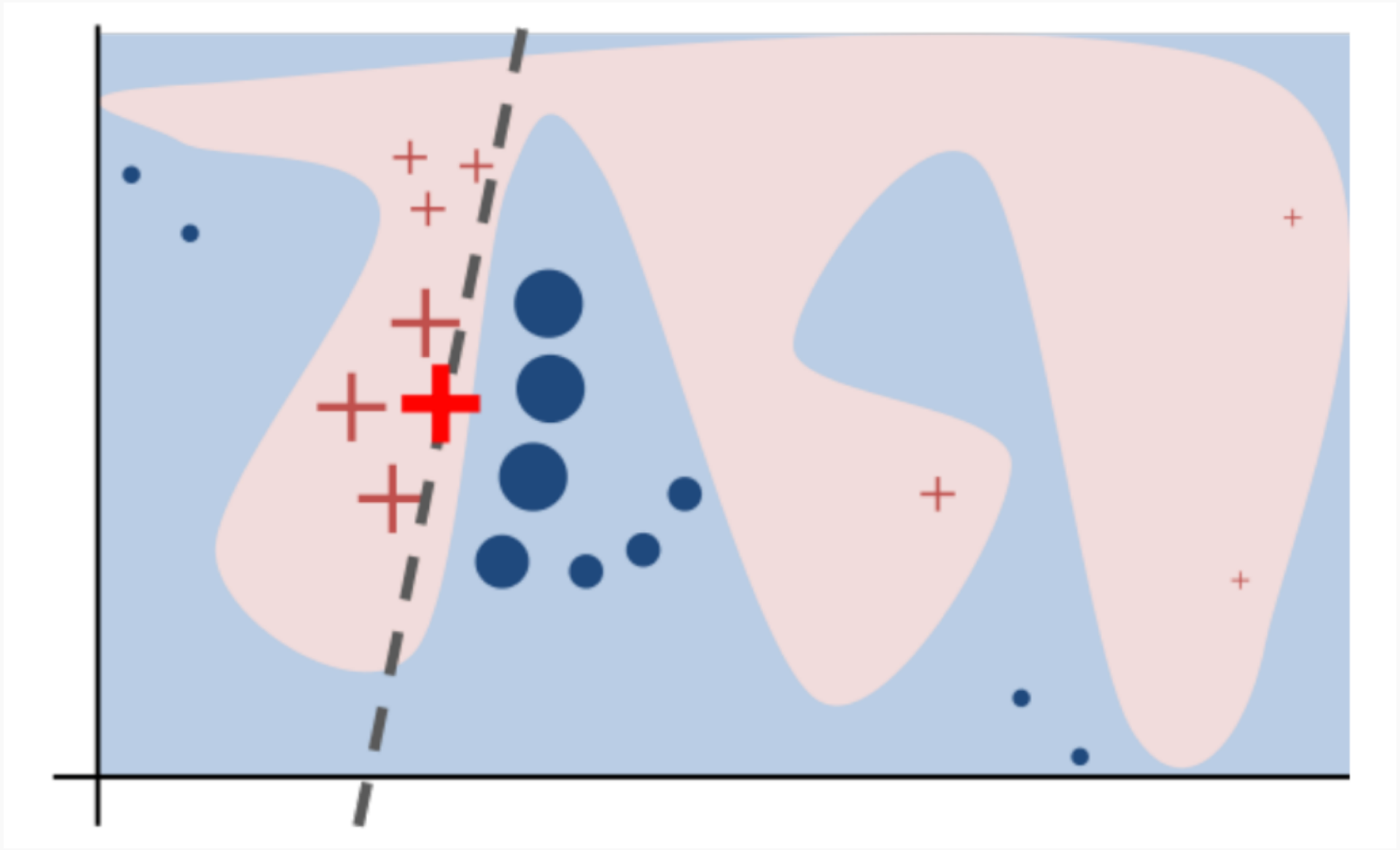
LIME: in a nutshell

The approach is to use a directly interpretable model (aka, linear regression) to help explain a model that is not directly interpretable.

To interpret how predictors are related to the response for a specific observation, synthetic data are created by perturbing that observations' predictors. Then the original model is used to create new *Y*s.

The interpretable model is then fit to the synthetic data, and the results are saved.

LIME: an illustrative picture



LIME: the algorithm

The 5-steps:

- **Select your observation of interest** for which you want to have an explanation of its black box prediction.
- **Perturb your observation** (rx: normal dist. perturbations for numerical predictors) and get the black box **predictions** for these new points.
- **Weight the new observations** based on their proximity to the instance of interest.
- **Train an interpretable local model (**linear regression with Ridge penalty**)** on the weighted, perturbed observations.
- Explain the prediction by **interpreting the local model**.

LIME: Pros and Cons

Advantages:

- Easy to apply LIME to tabular data, text data, and images
- They make **human-friendly** explanations.
- Comes with a **fidelity measure** that gives us a good idea of how reliable the interpretable model is in explaining the black box predictions.

Disadvantages:

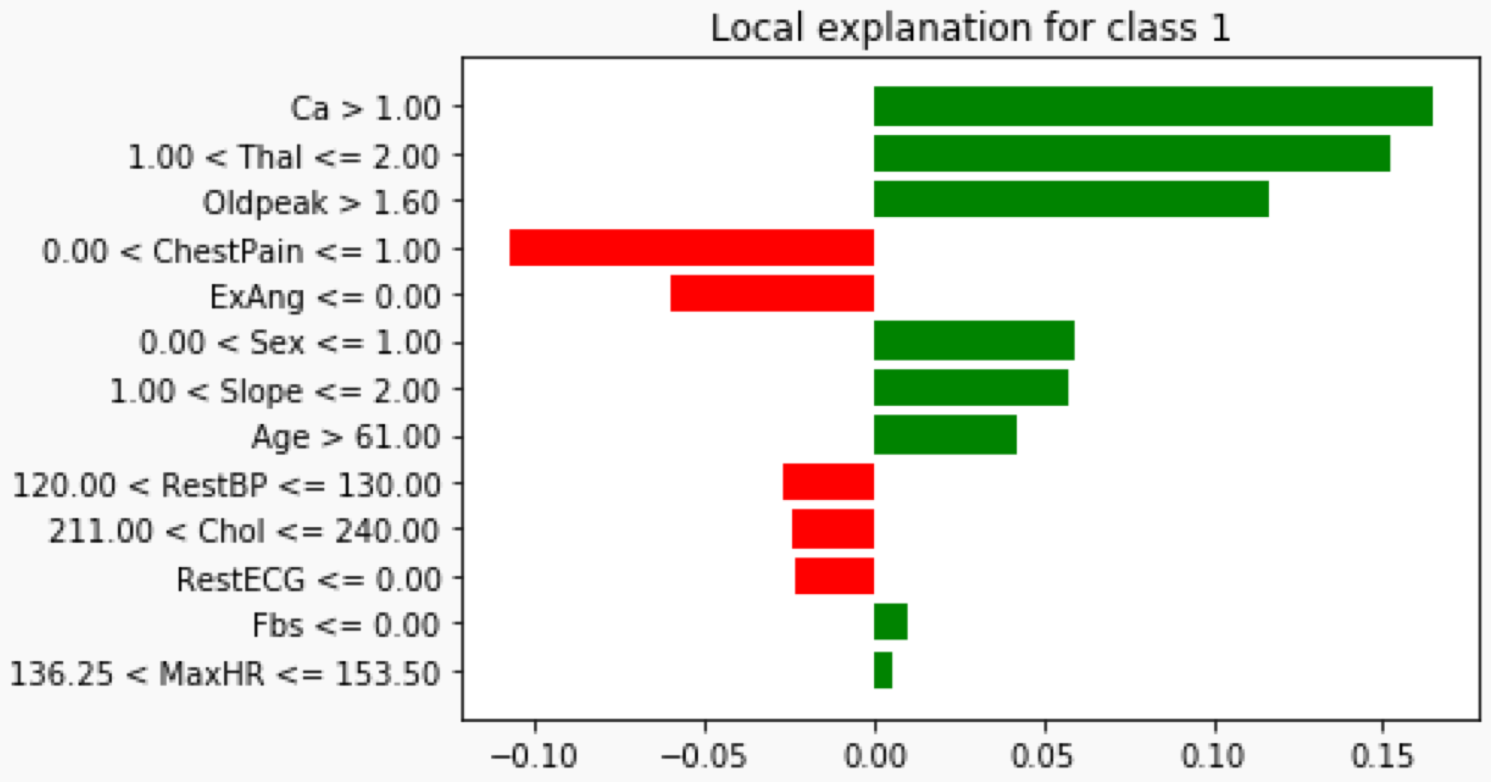
- The correct definition of the neighborhood is a very big, unsolved problem when using LIME with tabular data leading to **instability**.
- Interpretations are single **observation-specific**. Do not get easy generalities as to how *X* relates to *Y*.

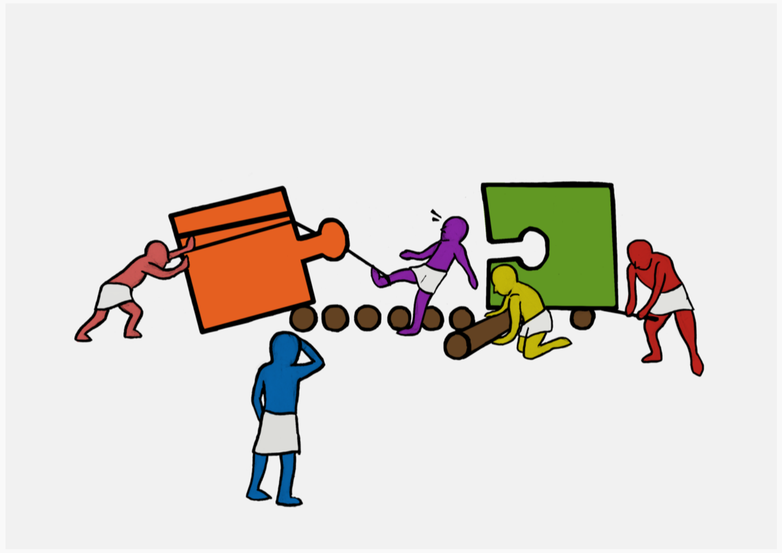
LIME: the equation

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \mathcal{L}(f, g, \Pi_x) + \Omega(g)$$

- x is a specific observation at which you want to do interpretations
- f is the machine learning model that is hard to explain
- G is the class of interpretable models you are considering using to explain f, and g is the specific model use estimate
- $\mathcal{L}$ is a measure for how well g approximates f
- Π_x is the locality/neighborhood of the observation being used around x
- $\Omega(g)$ is the restriction placed on g (number of non-zero entries, for example)
- $\xi(x)$ is the resulting interpretation of model f using g at x.

LIME: an output





Exercise Time!

<u>Exercise (graded): Interpreting Models</u>