Open the Black Box

Introduction to Model Interpretability

@KevinLemagnen
@cambridgespark

PYCON CLEVELAND 2019

CAMBRIDGE SPARK

Model Interpretability - Outline

1. Introduction - Why do we need it?

2. Eli5 and Permutation Importance

3. LIME - **L**ocal **I**nterpretable **M**odel-agnostic **E**xplanations

4. SHAP - **SH**apley **A**dditive ex**P**lanation

CAMBRIDGE SPARK

Model Interpretability:

Why do we need it?

CAMBRIDGE SPARK

Job Done!

- ✓ Cleaned and preprocessed **messy data**
- ✓ **Engineered** fancy new features
- ✓ Selected the **best model** and tuned hyperparameters
- ✓ Trained your **final model**
- ✓ Got **great performance** on the test set

CAMBRIDGE SPARK

Job Done . . . or not ...

CAMBRIDGE SPARK

Job Done … or not …

Can you explain how your model works?

CAMBRIDGE SPARK

Why is Interpretability important?

Algorithms are everywhere, sometimes automating important decisions that have an impact on people.

- **Insurance**: “predict the optimal price to charge a client”
- **Bank**: “predict who to give a loan to or not”
- **Police**: “predict who is more likely to commit a crime”
- **Social media**: “predict who is most likely to click on an ad”
- [...]

Black-box models are not an option

Build Trust in the Model

Goal: Predict employees' performance for a large organisation

Data: Performance reviews of individual employees for the last 10 years

What if that company tends to promote men more than women?

The model will **learn the bias,** and predict that male employees are more likely to perform better...

"Models are opinions embedded in mathematics" - Cathy O'Neil

CAMBRIDGE SPARK

Help Decision Maker

Goal: Predict the likelihood of a patient to develop a disease X

Data: Symptoms and information about past patients and whether they had X.

Here our model is meant to be used to inform doctors and help them with diagnosis.

Doctors need to understand exactly **why** the model thinks a patient has X before treating them.

Debugging the Model

Goal: Classify images - Wolves vs Huskies

Data: Pictures of wolves and huskies

What if pictures of wolves show something different in the background?

(a) Husky classified as wolf (b) Explanation

"Why Should I Trust You?": Explaining the Predictions of Any Classifier
Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin

CAMBRIDGE SPARK

Some models are easy to interpret

Linear/Logistic regression

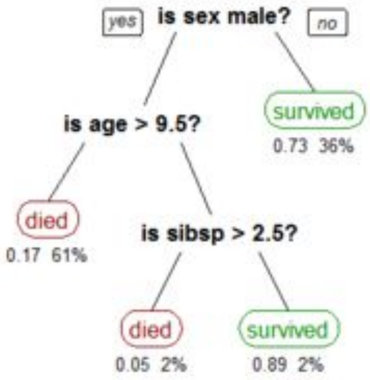
- Weight on each feature
- Know the exact contribution of each feature, negative or positive

$$Y = 3 * X1 - 2 * X2$$

Increasing X1 by 1 unit increases Y by 3 units

Single Decision Tree

- Easy to understand how a decision was made by reading from top to bottom

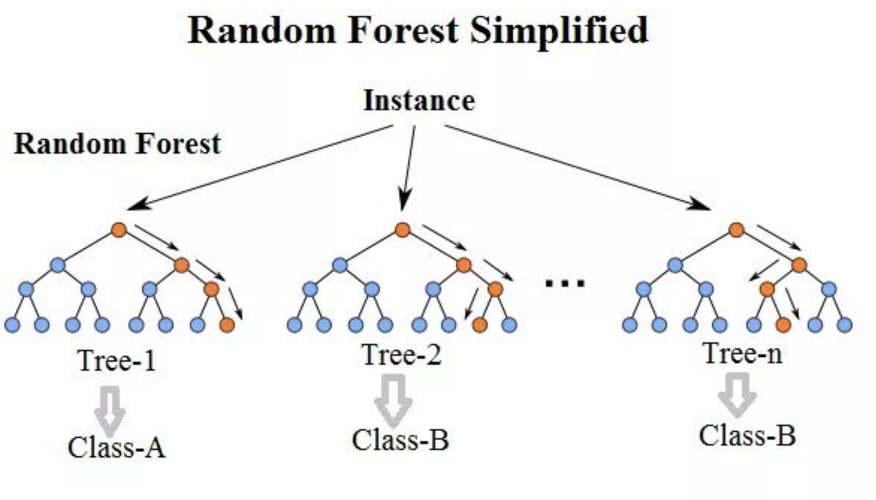


CAMBRIDGE SPARK

Some models are harder to interpret

Ensemble models (random forest, boosting, etc…)

- Hard to understand the role of each feature
- Usually comes with **feature importance**
- Doesn't tell us if feature affects decision positively or negatively

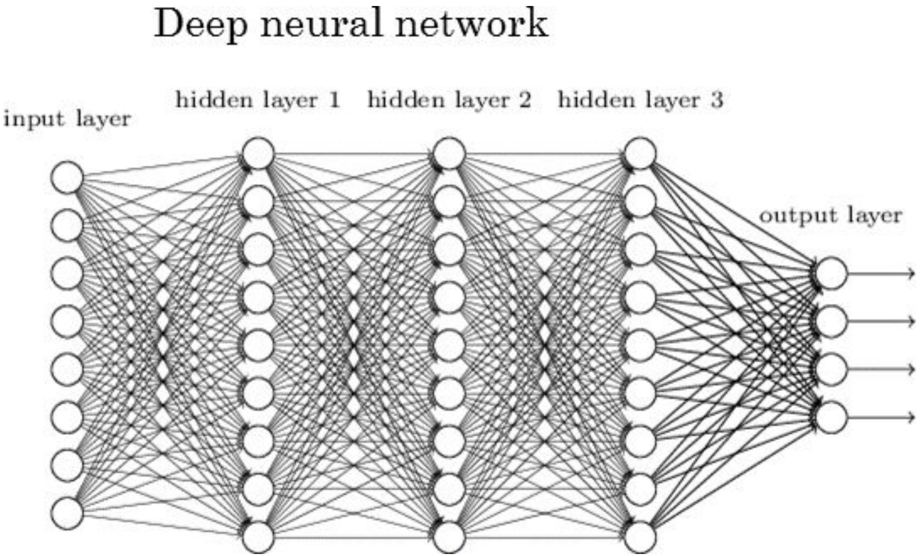


CAMBRIDGE SPARK

Some are really hard to interpret

Deep Neural Networks

- No straightforward way to relate **output** to **input** layer
- "Black-box"



CAMBRIDGE SPARK

Does it mean we can only use simple models?

- Sticking to **simple models** is the best way to ensure interpretability
    - Trade off Interpretability vs Performance
- Model agnostic techniques allow usage of complex models whilst maintaining interpretability

CAMBRIDGE SPARK

Different kinds of interpretation

- **Local**: Explain how and why a specific prediction was made
    - *Why did our model predict that patient X has disease Y*
- **Global:** Explain how our model works overall
    - *What symptoms are generally important for our model and how do they impact the outcome?*
    - Feature Importance is a global interpretation with amplitude only, no direction

CAMBRIDGE SPARK

ELI5 and Permutation Importance

"Explain Like I'm 5"

CAMBRIDGE SPARK

ELI5

- Can be used to interpret **sklearn**-like models
- Create nice visualisations for **white-box models**
    - Can be used for local and global interpretation
- Implements **Permutation Importance** for black-box models
    - Only global interpretation

ELI5 - for white-box models

Explain model **globally** (feature importance):

```
> eli5.show_weights(model)
```

Weight?	Feature
+2.117	mean radius
+1.276	texture error
+1.266	worst radius
+0.397	<BIAS>
+0.108	mean texture
… 3 more positive …	
… 8 more negative …	
-0.071	mean perimeter
-0.095	area error
-0.117	worst fractal dimension
-0.119	worst perimeter
-0.155	mean smoothness
-0.227	mean symmetry
-0.287	worst smoothness
-0.337	worst texture
-0.342	mean concave points
-0.407	mean compactness
-0.649	mean concavity
-0.659	worst concave points
-0.697	worst symmetry
-1.159	worst compactness
-1.603	worst concavity

ELI5 - for white-box models

Explain model **locally**:

```
> eli5.show_prediction(model, observation)
```

y=0 (probability **0.797**, score **-1.371**) top features

Contribution?	Feature
+17.212	worst area
+12.967	worst perimeter
+10.803	worst texture
+6.671	mean perimeter
+1.828	area error
+1.538	mean area
+1.113	worst concavity
+0.895	worst compactness
+0.251	worst symmetry
+0.146	worst concave points
+0.138	mean concavity
+0.093	mean compactness
+0.047	worst smoothness
+0.047	mean symmetry
+0.027	mean concave points
+0.018	mean smoothness
+0.017	worst fractal dimension
+0.005	radius error
+0.003	concavity error
+0.002	mean fractal dimension
+0.001	symmetry error
+0.001	concave points error
+0.000	smoothness error
-0.000	fractal dimension error
-0.000	compactness error
-0.035	perimeter error
-0.397	<BIAS>
-1.491	texture error
-2.432	mean texture
-19.026	worst radius
-29.072	mean radius

ELI5 - Permutation Importance

- Model Agnostic
- Provides **global** interpretation
  - Only amplitude, do not specify in what way it impacts the outcome

ELI5 - Permutation Importance

For each feature:

1. **Shuffle** values in the provided dataset
2. Generate **predictions** using the model on the modified dataset
3. Compute the decrease in **accuracy** vs before shuffling

*We can compare the **impact on accuracy** of shuffling each feature individually.*

CAMBRIDGE SPARK

ELI5 - Permutation Importance

```
> perm = PermutationImportance(model)
> perm.fit(X, y)
> eli5.show_weights(perm)
```

Weight	Feature
0.4815 ± 0.0400	worst area
0.1779 ± 0.0318	worst perimeter
0.0931 ± 0.0214	mean radius
0.0903 ± 0.0148	area error
0.0707 ± 0.0134	worst radius
0.0615 ± 0.0157	worst texture
0.0450 ± 0.0108	mean perimeter
0.0102 ± 0.0060	mean area
0.0067 ± 0.0052	worst concavity
0.0053 ± 0.0039	worst compactness
0.0039 ± 0.0068	texture error
0.0021 ± 0.0026	worst symmetry
0.0004 ± 0.0014	worst concave points
0.0004 ± 0.0014	mean concave points
0.0004 ± 0.0014	mean concavity
0 ± 0.0000	mean symmetry
0 ± 0.0000	mean smoothness
0 ± 0.0000	mean compactness
0 ± 0.0000	mean fractal dimension
0 ± 0.0000	radius error

… 10 more …

CAMBRIDGE SPARK

Hands-on session

>>> ELI5

CAMBRIDGE SPARK

Interpretability - LIME

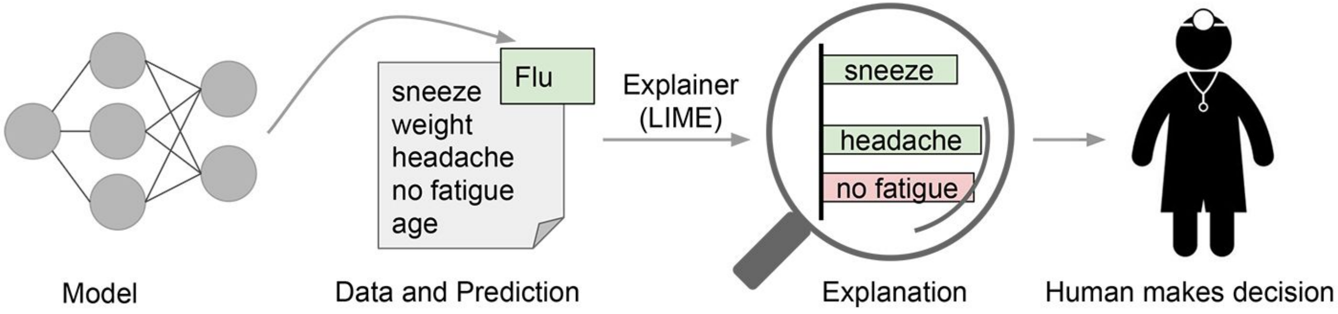
CAMBRIDGE SPARK

LIME - Local Interpretable Model-Agnostic Explanations

Local: Explains why a single data point was classified as a specific class

Model-agnostic: Treats the model as a black-box. Doesn't need to know *how* it makes predictions

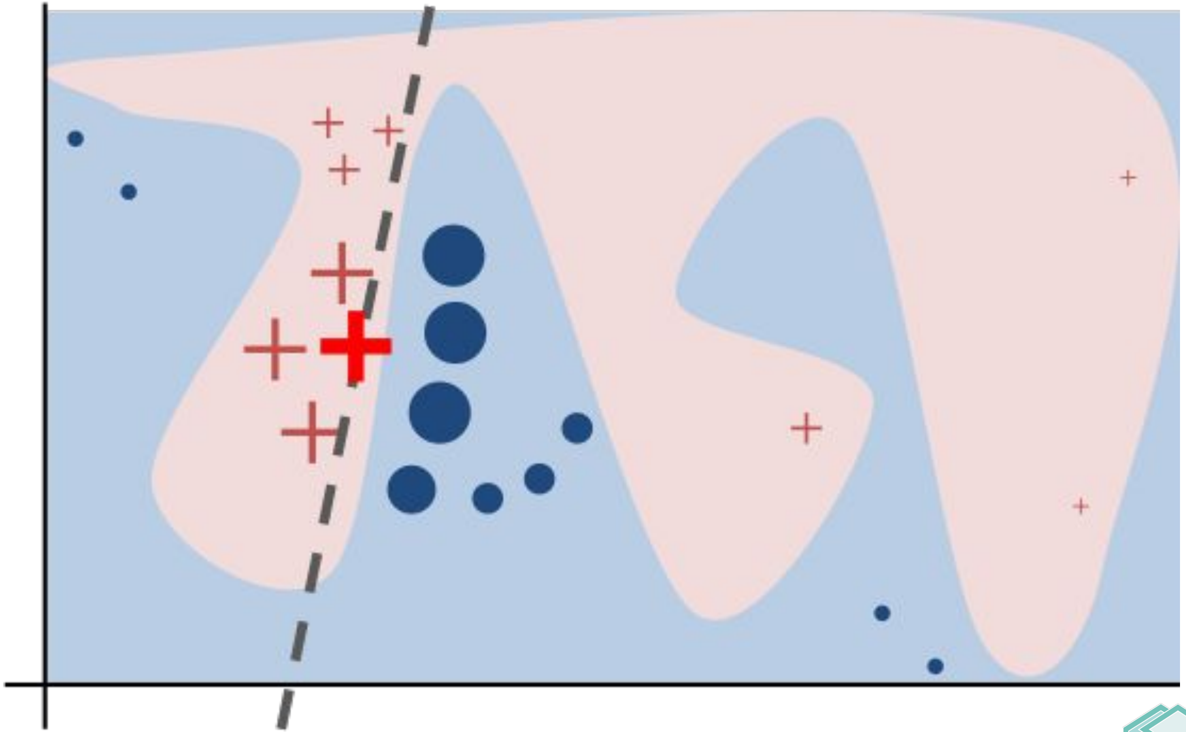
Paper "*Why should I trust you?*" published in August 2016.



"Why Should I Trust You?": Explaining the Predictions of Any Classifier
Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin

CAMBRIDGE SPARK

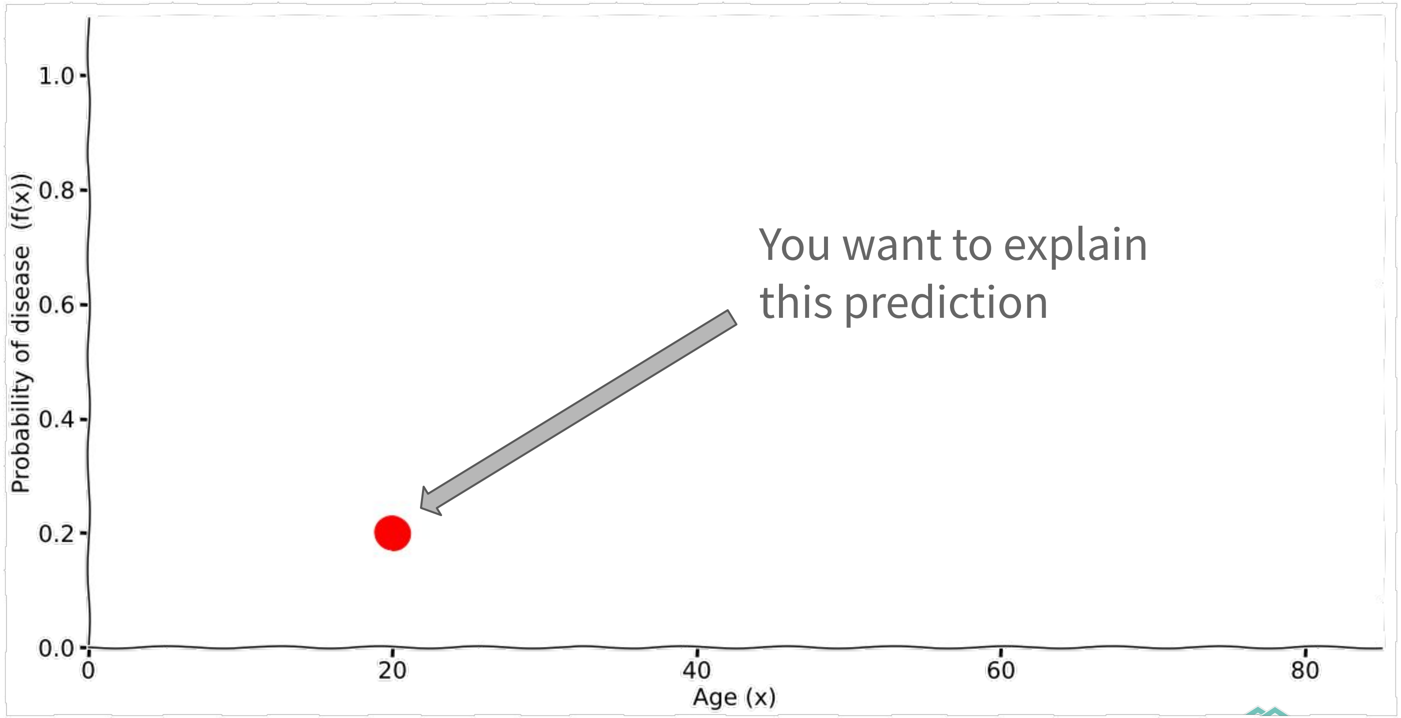
LIME - How does it work? (simplified version)



"Why Should I Trust You?": Explaining the Predictions of Any Classifier
Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin

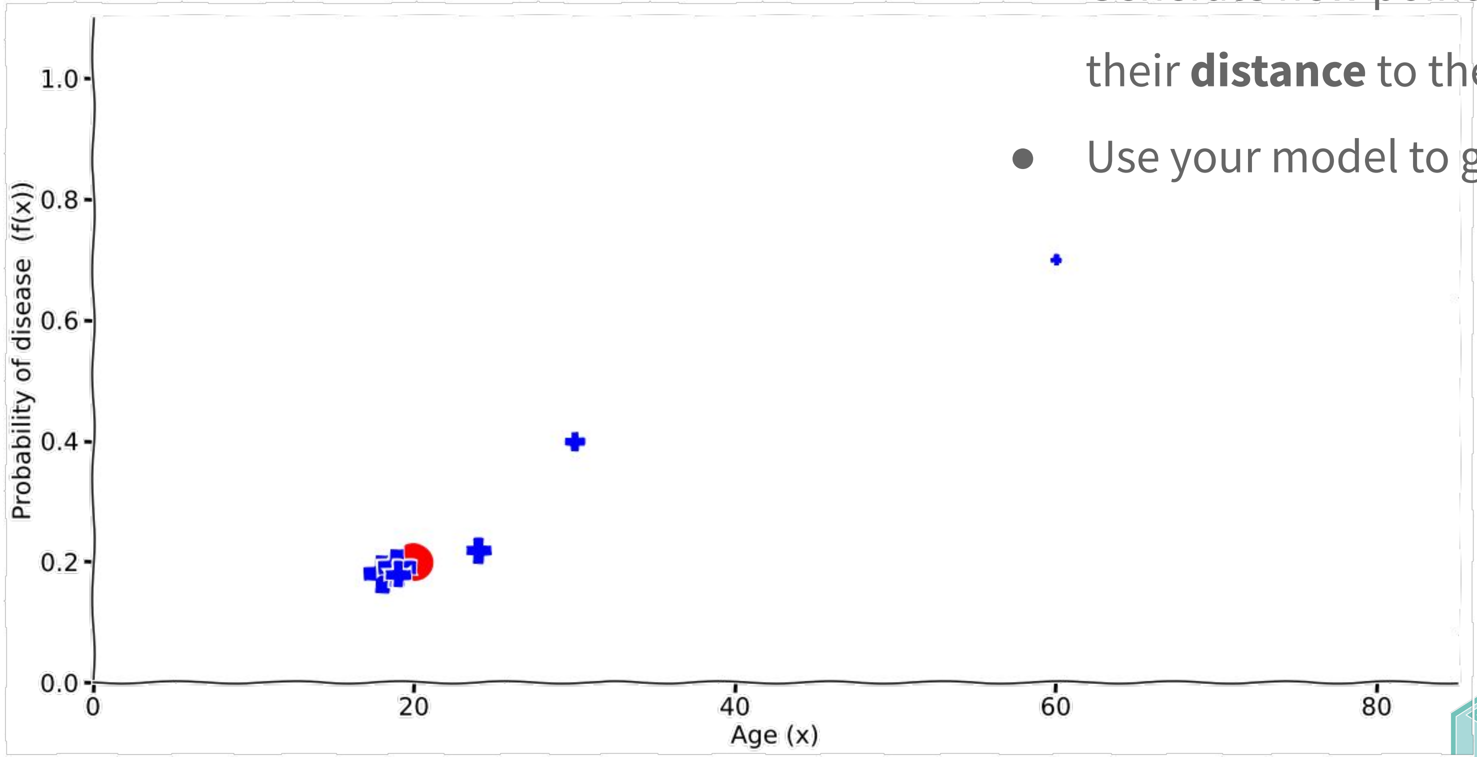
CAMBRIDGE SPARK

LIME - How does it work?



CAMBRIDGE SPARK

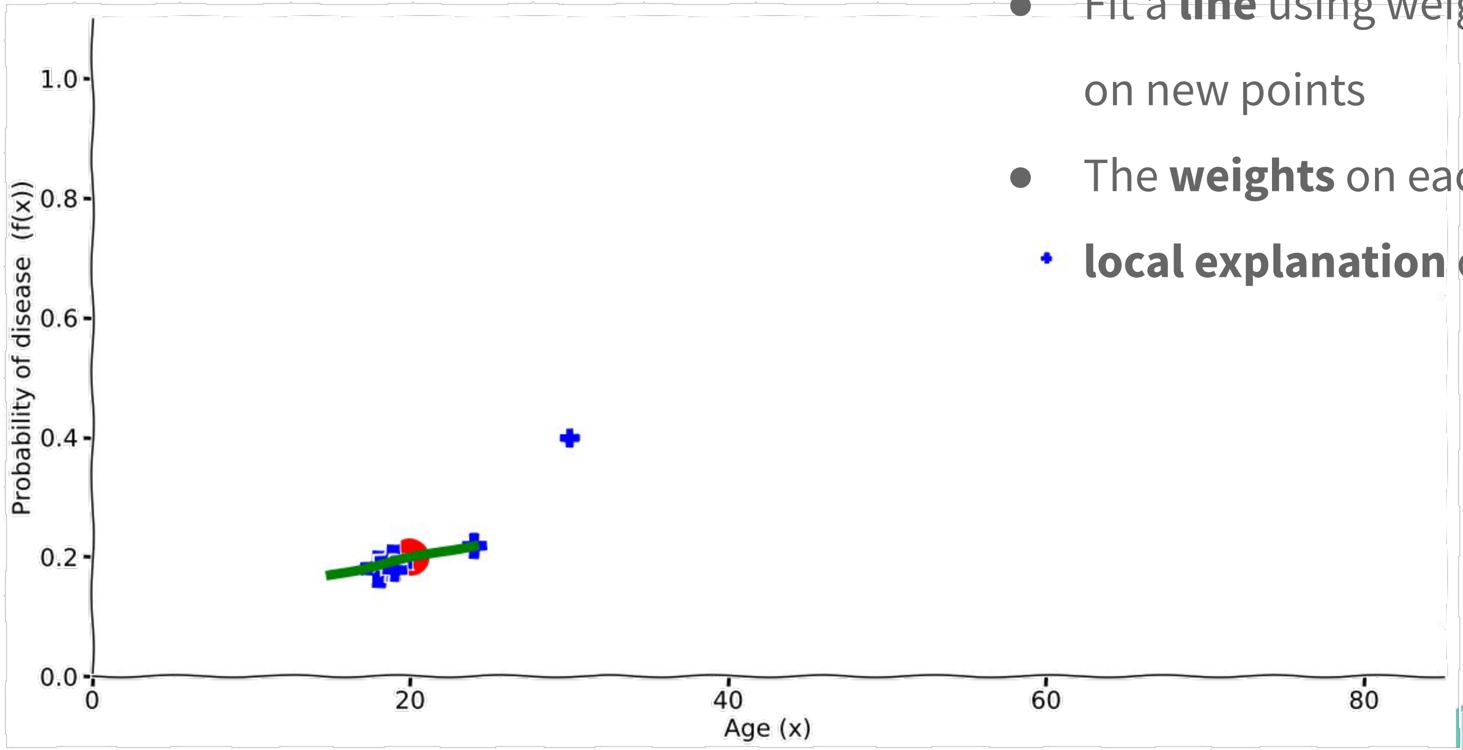
LIME - How does it work?



- Generate new points, weighted by their **distance** to the observation
- Use your model to get **f(x)**

CAMBRIDGE SPARK

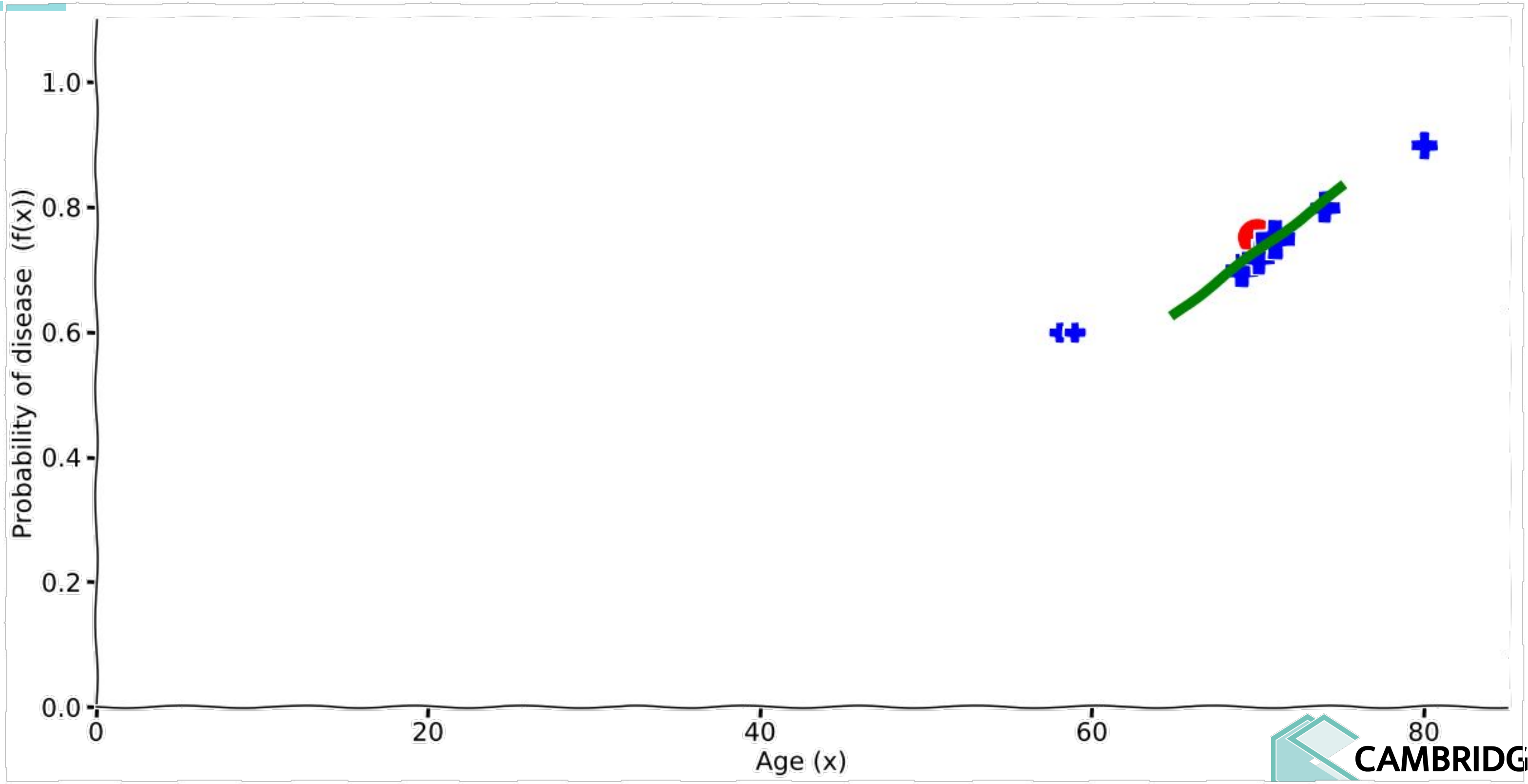
LIME - How does it work?



- Fit a **line** using weighted least square on new points
- The **weights** on each feature give a **local explanation** of the model

CAMBRIDGE SPARK

LIME - How does it work?



CAMBRIDGE SPARK

LIME - How does it work?

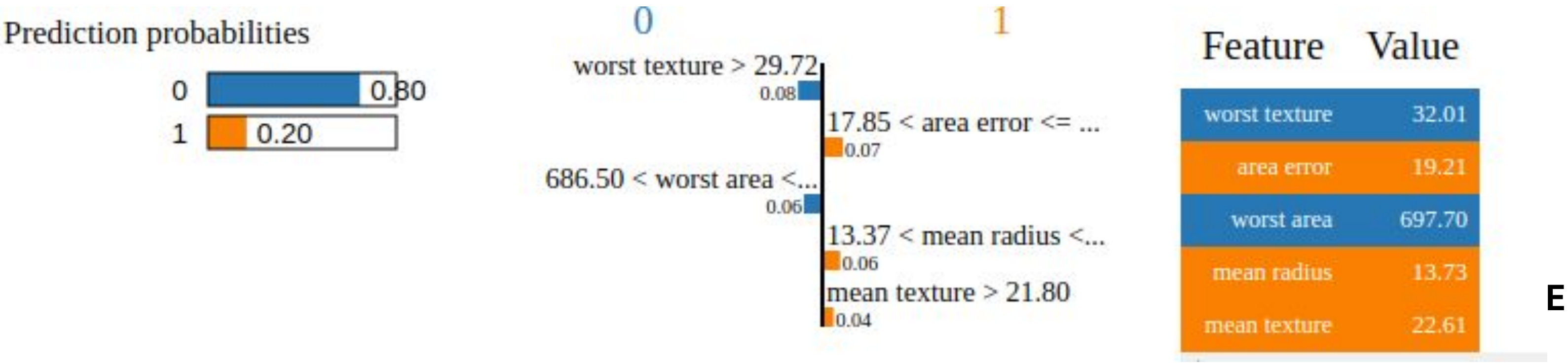
Pick an **observation** to explain and a number **m** of features to use, then:

1. Create **new dataset** around observation by sampling from distribution learnt on training data
2. Calculate distances between points and **observation**, that's our measure of similarity
3. Use model to predict **probability** on new points, that's our new **y**
4. Uses **feature selection** to find the **m** features that have the strongest relationship with our target
5. Fit a linear model on data in **m** dimensions weighted by similarity
6. **Weights** of linear model are used as explanation of decision

CAMBRIDGE SPARK

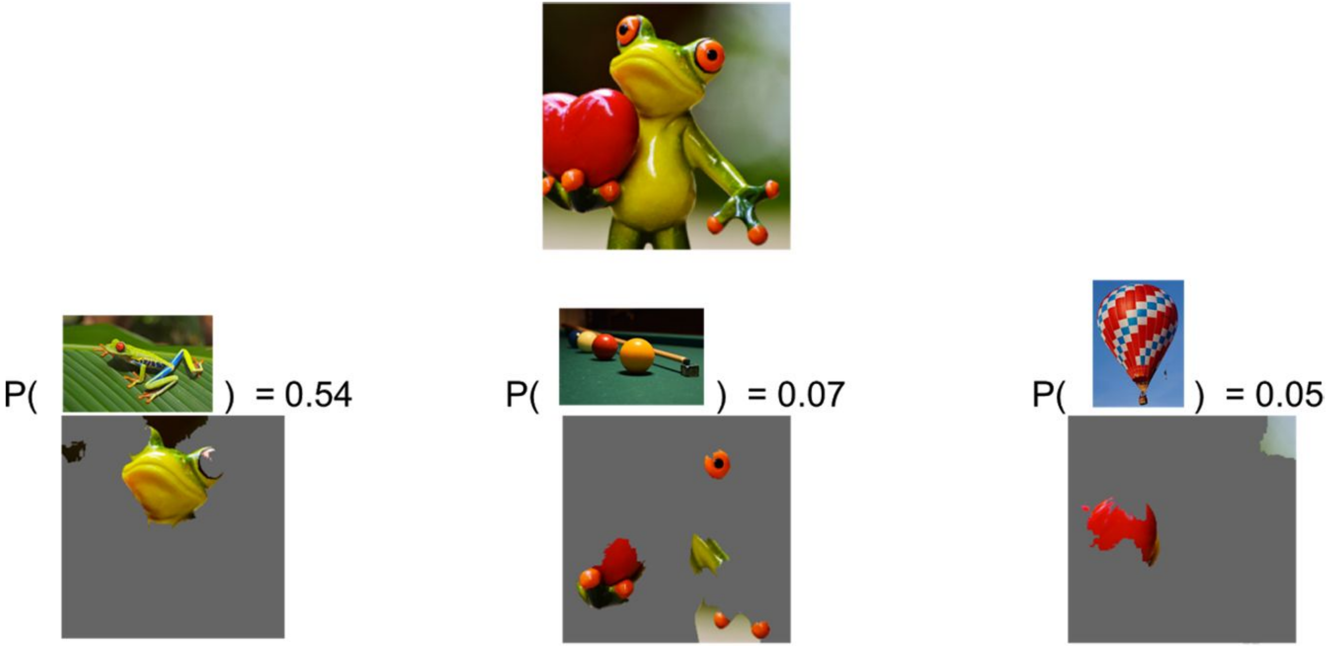
LIME - Explanation

- Central plot shows **importance** of the top features for this prediction
  - Value corresponds to the **weight** of the linear model
  - **Direction** corresponds to whether it pushes in one way or the other
- Numerical features are **discretized** into bins
  - Easier to interpret: weight == contribution / sign of weight == direction



E SPARK

LIME - Can be used on images too



"Why Should I Trust You?": Explaining the Predictions of Any Classifier
Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin

CAMBRIDGE SPARK

LIME - Some drawbacks

- Depends on the **random sampling** of new points, so it can be unstable
- Fit of linear model can be **inaccurate**
  - But we can check the r-squared score to know if that's the case
- Relatively slow for a single observation, in particular with images

LIME - Available “Explainers”

Lime supports many types of data:

- Tabular Explainer
- Recurrent Tabular Explainer
- Image Explainer
- Text Explainer

CAMBRIDGE SPARK

LIME - API

1. **Create a new explainer from dataset**
   > `my_explainer = Explainer(data)`
2. **Select an observation and create an explanation for it**
   > `observation = np.array([...])`
   > `my_explanation = explainer.explain_instance(`*`observation`*`,`
   > *`predict_function`*`,`
   > `num_features=5)`
3. **Use methods on explanation to visualise results**
   > `my_explanation.show_in_notebook()`
   > `my_explanation.get_image_and_mask()`
   > `[...]`

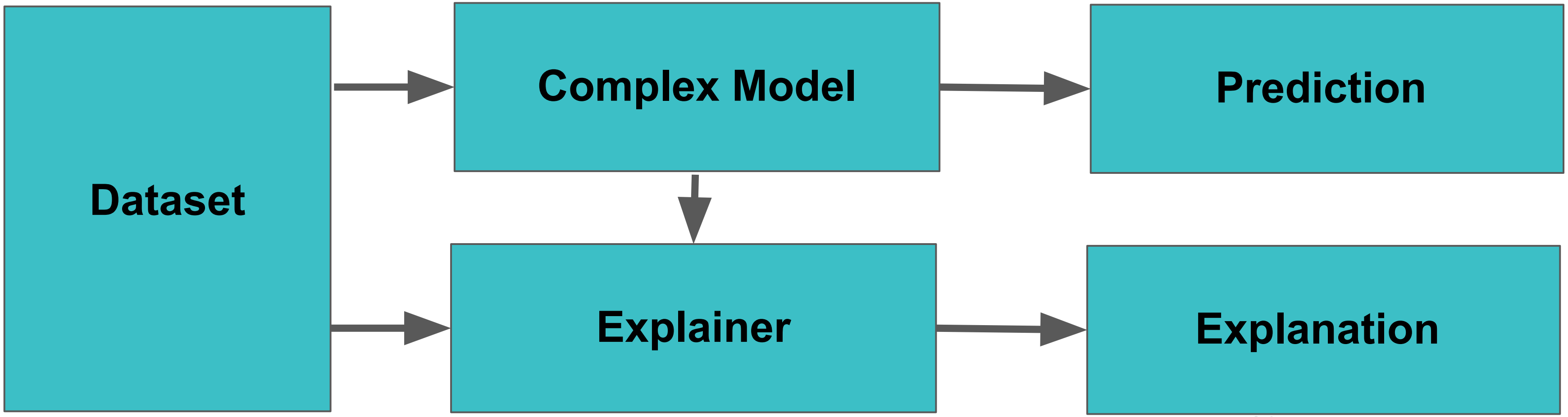
Hands-on session

\>>> LIME

Interpretability - SHAP

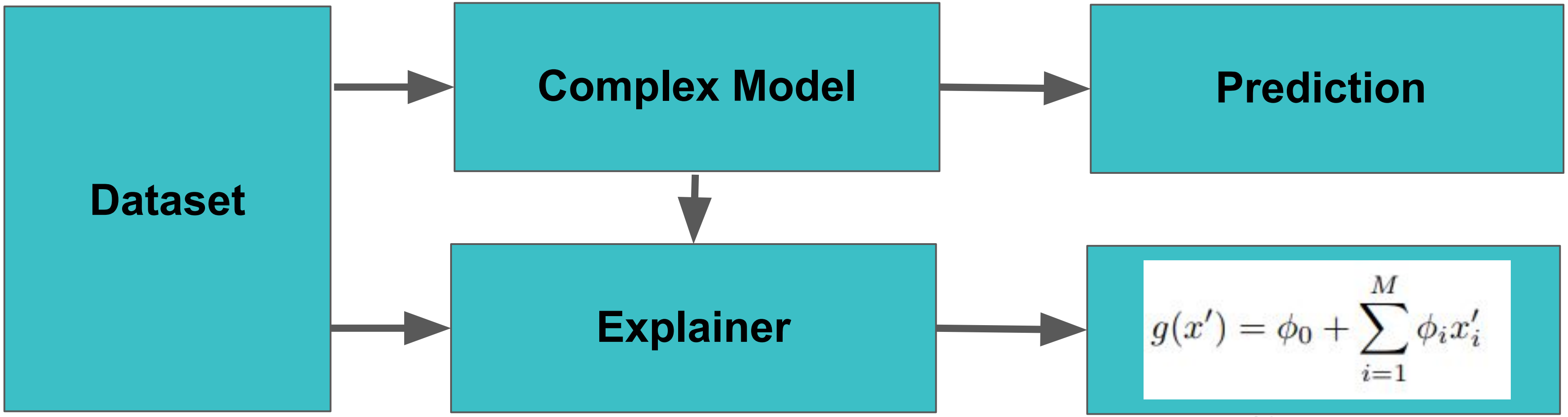
Shapley Additive Explanations (SHAP)

- An explanation model is a simpler model that is a good approximation of our complex model.



Shapley Additive Explanations (SHAP)

- An explanation model is a simpler model that is a good approximation of our complex model.
- “Additive feature attribution methods”: the local explanation is a linear combination of the features.
    - Weights are the SHAP values



Shapley Additive Explanations (SHAP)

For a given observation, we compute a **SHAP value** per feature, such that:

sum(**SHAP_values_for_obs**) = **prediction_for_obs** - **model_expected_value**

- **SHAP values** are in same unit as prediction/model expected value (probability, log odds, etc…)
- Model's **expected value** is the average prediction made by our model

CAMBRIDGE SPARK

Shapley Additive Explanations (SHAP)

For a given observation, we compute a **SHAP value** per feature, such that:

sum(**SHAP_values_for_obs**) = **prediction_for_obs** - **model_expected_value**

Interpretation: SHAP values correspond to the contribution of each feature towards "pushing" the prediction away from the expected value

CAMBRIDGE SPARK

Shapley Additive Explanations (SHAP)

For the **model agnostic** explainer, SHAP leverages Shapley values from Game Theory.

To get the importance of feature **X{i}**:

- Get all subsets of features **S** that do not contain **X{i}**
- Compute the effect on our predictions of adding **X{i}** to all those subsets
- Aggregate all contributions to compute **marginal contribution** of the feature

Shapley Additive Explanations (SHAP)

- To estimate **expected_value**, we need to provide some training data
- "**Missing feature**" is approximated by setting its value to the expected value of the feature learnt on the training data

Simulating all combinations of features, for each individual feature is **computationally expensive**.

Optimised versions for specific classes of models (trees, linear, neural networks, …)

CAMBRIDGE SPARK

Shapley Additive Explanations (SHAP)

With the Tree Explainer:

- No need to provide a background dataset
- The model's **expected_value** is known from the trained model
- Contributions of each individual features can be **computed directly** from the structure of the tree

Shapley Additive Explanations (SHAP)

- **TreeExplainer**
  - For tree based models
  - Works with scikit-learn, xgboost, lightgbm, catboost
- **DeepExplainer**
  - For Deep Learning models
- **KernelExplainer**
  - Model agnostic explainer

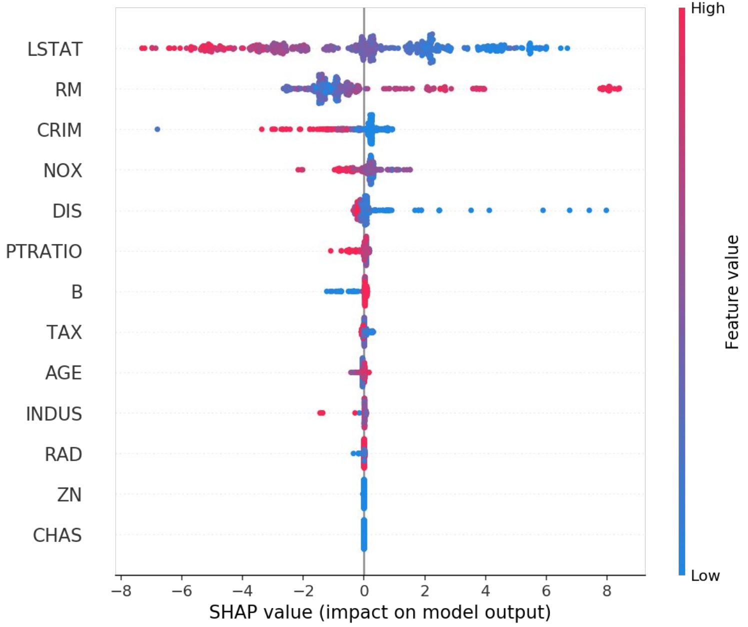
SHAP - Local Interpretation

- Base value is the expected value of your model
- Output value is what your model predicted for this observation
- In red are the positive SHAP values, the features that contribute positively to the outcome
- In blue the negative SHAP values, features that contribute negatively to the outcome

CAMBRIDGE SPARK

SHAP - Global Interpretation

- Although SHAP values are local, by plotting all of them at once we can learn a lot about our model globally



CAMBRIDGE SPARK

SHAP - Tree Explainer API

1. Create a new explainer, with our model as argument
   - `explainer = TreeExplainer(my_tree_model)`
2. Calculate shap_values from our model using some observations
   - `shap_values = explainer.shap_values(observations)`
3. Use SHAP visualisation functions with our shap_values
   - `shap.force_plot(base_value, shap_values[0]) # Local explanation`
   - `shap.summary_plot(shap_values) # Global features importance`

CAMBRIDGE SPARK

Hands-on session

>>> SHAP

CAMBRIDGE SPARK

Conclusion

- Gives trust that our complex model predicts the right thing
- Can help debugging our model and spot biases in our data
- Can explain to others why a prediction was made
- Regulations make it mandatory (finance, GDPR, …)