Data Science in the Wild

Lecture 14: Explaining Models

Eran Toch

CORNELL TECH

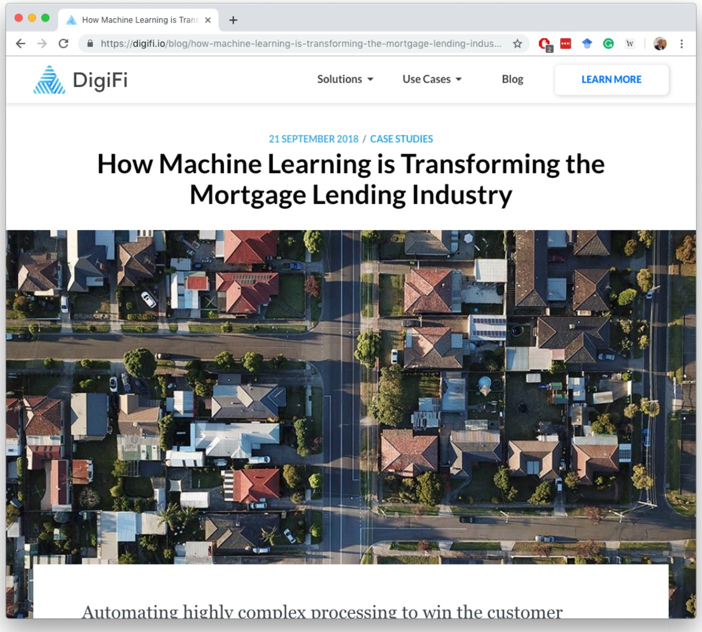
Agenda

1. Explaining models
2. Transparent model explanations
3. Obscure model explanations
4. LIME: Local Interpretable Model-Agnostic Explanations

Models and their power

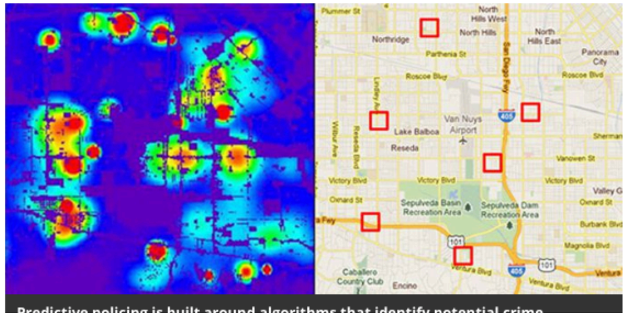
Accelerating the discovery of novel immuno-oncology targets

Interpretation of big data in the context of the entire corpus of knowledge is a challenge. AI gives us the ability to do so in a consistent and reproducible way.



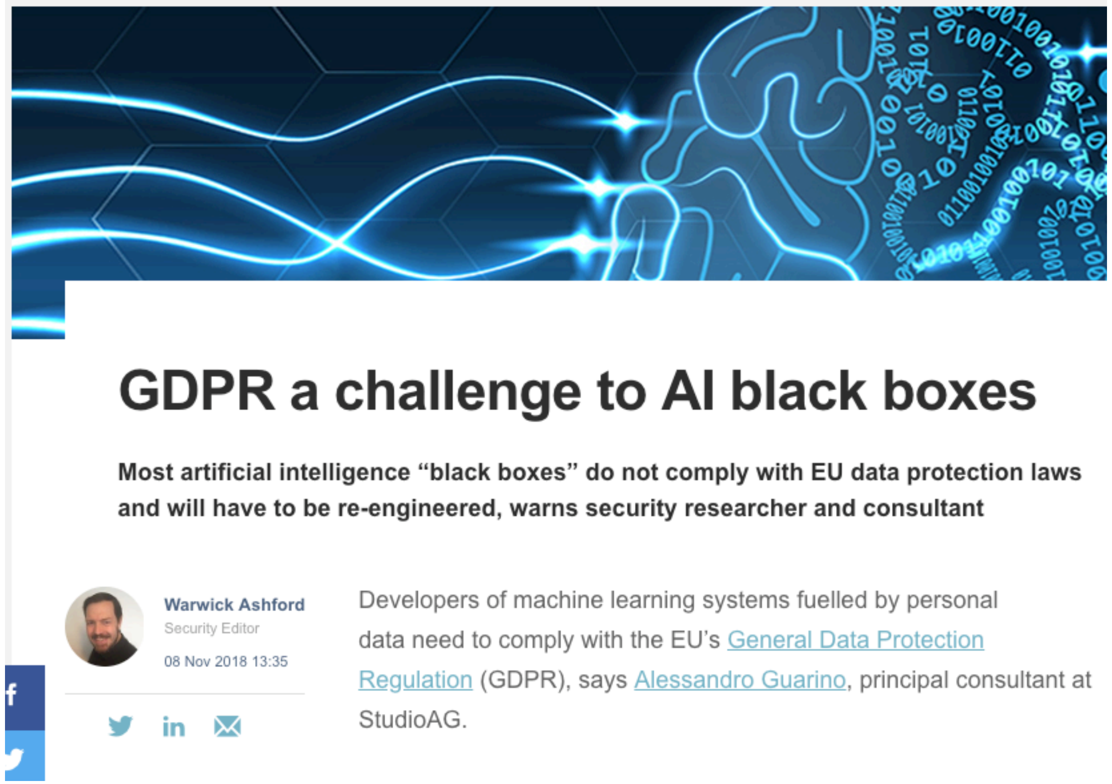
Artificial Intelligence Is Now Used to Predict Crime. But Is It Biased?

The software is supposed to make policing more fair and accountable. But critics say it still has a way to go.



We do we need to explain models

- Scaling models beyond particular datasets
- Providing intuitive explanations and generating human-understandable models
- Legal requirements (GDPR) and Cal law
- Identifying bias



GDPR a challenge to AI black boxes

Most artificial intelligence "black boxes" do not comply with EU data protection laws and will have to be re-engineered, warns security researcher and consultant

Warwick Ashford
Security Editor
08 Nov 2018 13:35

Developers of machine learning systems fuelled by personal data need to comply with the EU's General Data Protection Regulation (GDPR), says Alessandro Guarino, principal consultant at StudioAG.

Example: scaling models

- Classifying images to husky dogs versus wolves
- We classifies the images with 90% accuracy
- But, can It scale?

(a) Husky classified as wolf (b) Explanation

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier." *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016.

What is Interpretability?

- Definition Interpret means to explain or to present in understandable terms
- In the context of ML systems, we define interpretability as the ability to explain or to present in understandable terms to a human

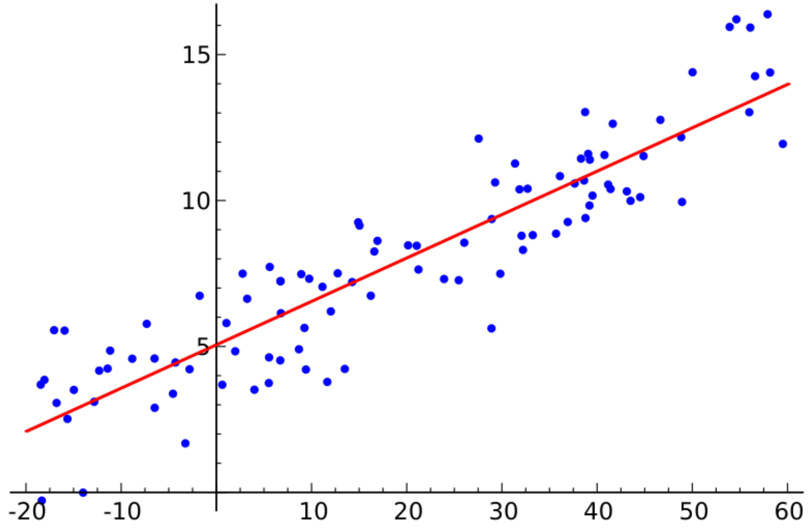
Towards A Rigorous Science of Interpretable Machine Learning Finale Doshi-Velez and Been Kim

White Box Explanations

Existing explainable models: Linear/Logistic regression

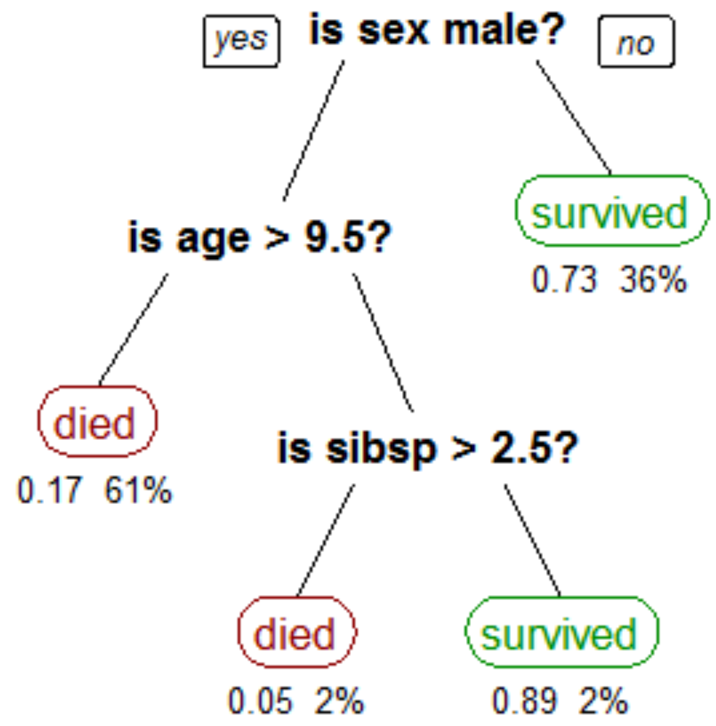
$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\mathsf{T}\boldsymbol\beta + \varepsilon_i, \qquad i = 1, \ldots, n,$$

- Each feature has a weight
- We can calculate the contribution of each feature, individually (under some reasonable assumptions) to the dependent variable



Existing explainable models: Single decision trees

- A single decision tree provides a hierarchical explanation model
- Easy to understand and to operationalize



ELI5

- Explain Like I'm 5
- Useful to debug sklearn models and communicate with domain experts
- Provides global interpretation of transparent models with a consistent API
- Provides local explanation of predictions

Example

- The data is related with direct marketing campaigns of a Portuguese banking institution
- 41188 records and 20 features
- Predict whether or not the client targeted by the campaign ended up subscribing

S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

Input variables:
bank client data:
1 - age (numeric)
2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')
6 - housing: has housing loan? (categorical: 'no','yes','unknown')
7 - loan: has personal loan? (categorical: 'no','yes','unknown')
related with the last contact of the current campaign:
8 - contact: contact communication type (categorical: 'cellular','telephone')
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.
other attributes:
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - previous: number of contacts performed before this campaign and for this client (numeric)
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
social and economic context attributes
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
17 - cons.price.idx: consumer price index - monthly indicator (numeric)
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):
21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

Logistic regression models

```
# Logistic Regression
lr_model = Pipeline([("preprocessor", preprocessor),
                     ("model", LogisticRegression(class_weight="balanced", solver="liblinear",
random_state=42))])
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=.3, random_state=42)

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=.3, random_state=42)

lr_model.fit(X_train, y_train)
y_pred = lr_model.predict(X_test)
accuracy_score(y_test, y_pred)

0.8323217609452133

print(classification_report(y_test, y_pred))
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.86      0.90     10965
           1       0.36      0.65      0.46      1392

   micro avg       0.83      0.83      0.83     12357
   macro avg       0.66      0.75      0.68     12357
weighted avg       0.88      0.83      0.85     12357
```

https://github.com/klemag/pydata_nyc2018-intro-to-model-interpretability

ELI5

```
import eli5
eli5.show_weights(lr_model.named_steps[
"model"])
```

- eli5.show_weights(lr_model.named_steps ["model"], feature_names=all_features)
-

y=1 top features

Weight?	Feature
+1.033	x49
+0.707	x7
+0.607	x5
+0.575	x29
+0.397	x24
+0.370	x14
+0.308	x46
+0.280	x45
+0.241	x42
+0.210	x61
+0.170	x47
… 10 more positive …	
… 33 more negative …	
-0.168	x22
-0.193	x21
-0.195	x30
-0.280	x43
-0.280	x59
-0.333	x53
-0.606	x50
-0.626	x51
-0.894	x4

y=1 top features

Weight?	Feature
+1.033	month__mar
+0.707	euribor3m
+0.607	cons.price.idx
+0.575	education__illiterate
+0.397	marital__unknown
+0.370	job__retired
+0.308	month__dec
+0.280	month__aug
+0.241	contact__cellular
+0.210	poutcome__success
+0.170	month__jul
… 10 more positive …	
… 33 more negative …	
-0.168	marital__married
-0.193	marital__divorced
-0.195	education__professional.course
-0.280	contact__telephone
-0.280	poutcome__failure
-0.333	month__sep
-0.606	month__may
-0.626	month__nov
-0.894	emp.var.rate

Explain instances

```
i = 4
X_test.iloc[[i]]
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome
39993	27	unknown	single	university.degree	no	yes	no	cellular	jun	wed	665	4	3	2	success

```
eli5.show_prediction(lr_model.named_steps["model"],

lr_model.named_steps["preprocessor"].transform(X_te
st)[i],
                    feature_names=all_features,
show_feature_values=True)
```

y=1 (probability **0.963**, score **3.260**) top features

Contribution?	Feature	Value
+57.065	cons.price.idx	94.055
+1.519	emp.var.rate	-1.700
+0.542	euribor3m	0.767
+0.304	cons.conf.idx	-39.800
+0.241	contact__cellular	1.000
+0.210	poutcome__success	1.000
+0.122	day_of_week__wed	1.000
+0.117	default__no	1.000
+0.068	job__unknown	1.000
-0.004	pdays	3.000
-0.023	age	27.000
-0.037	education__university.degree	1.000
-0.039	loan__no	1.000
-0.039	<BIAS>	1.000
-0.040	housing__yes	1.000
-0.075	marital__single	1.000
-0.132	month__jun	1.000
-0.173	campaign	4.000
-0.297	previous	2.000
-56.067	nr.employed	4991.600

Decision Trees

- For Decision Trees, ELI5 only gives feature importance, which does not say in what direction a feature impact the predicted outcome

```
gs = GridSearchCV(dt_model, {"model__max_depth": [3, 5, 7],
                             "model__min_samples_split": [2, 5]},
                  n_jobs=-1, cv=5, scoring="accuracy")

gs.fit(X_train, y_train)
accuracy_score(y_test, y_pred)
0.8553046856033018
eli5.show_weights(dt_model.named_steps["model"],
feature_names=all_features)
```

Weight	Feature
0.7088	nr.employed
0.1340	cons.conf.idx
0.0444	cons.price.idx
0.0338	pdays
0.0238	euribor3m
0.0211	month__oct
0.0125	default__unknown
0.0081	poutcome__failure
0.0045	contact__telephone
0.0039	campaign
0.0031	age
0.0007	job__unknown
0.0005	day_of_week__mon
0.0005	education__unknown
0.0003	previous
0	marital__divorced
0	job__unemployed
0	education__basic.4y
0	marital__unknown
0	marital__single
	... 42 more ...

Contribution to outcome

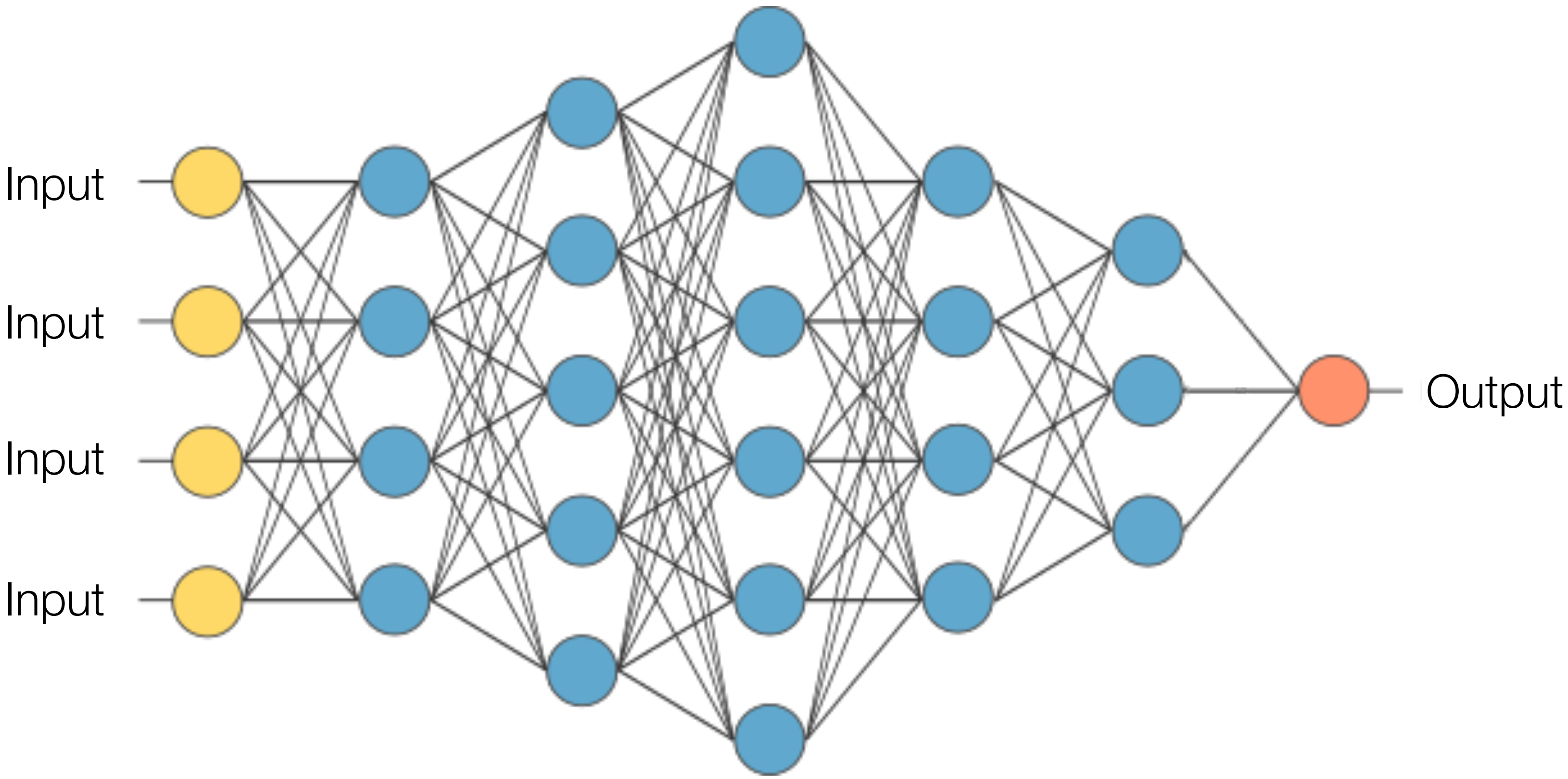
```
eli5.show_prediction(dt_model.named_steps["model"],
                     dt_model.named_steps["preprocessor"].transform(X_test)[i],
                     feature_names=all_features, show_feature_values=True)
```

y=0 (probability **0.758**) top features

Contribution?	Feature	Value
+0.500	<BIAS>	1.000
+0.137	nr.employed	5228.100
+0.097	cons.price.idx	94.465
+0.042	cons.conf.idx	-41.800
+0.014	age	35.000
-0.032	euribor3m	4.947

Obscure Box Explanations

Obscure Models

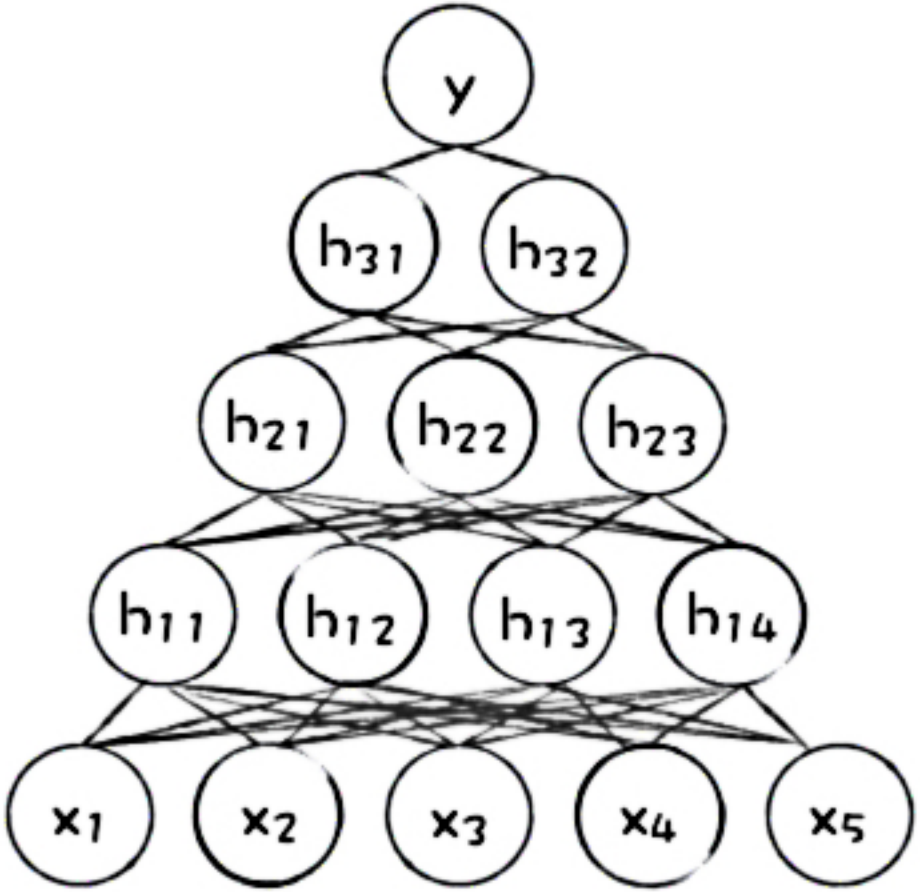


Good explainable models

- **Interpretable**: provide qualitative understanding between the input variables and the response
- **Local fidelity**: , for an explanation to be meaningful it must at least be locally faithful, i.e. it must correspond to how the model behaves in the vicinity of the instance being predicted
- **Model-agnostic**: an explainer should be able to explain any model
- **Global perspective**: Select a few explanations to present to the user, such that they are representative of the model

Hard in the general case

- Complex ML models learn from high-degree interactions between input variables
- For example, in a deep neural network, the original input variables X1-X5 are combined in the next level
- It is hard to portray the relationship between X1-X5 and Y



https://www.oreilly.com/ideas/testing-machine-learning-interpretability-techniques

The Multitude of Good Models

- Complex machine learning algorithms can produce multiple accurate models with very similar, but not the exact same, internal architectures
- Each of these different weightings would create a different function for making loan default decisions, and each of these different functions would have different explanations

Picture 1

$$y = 2.1 + 3.8x_3 - 0.6x_8 + 83.2x_{12} - 2.1x_{17} + 3.2x_{27},$$

Picture 2

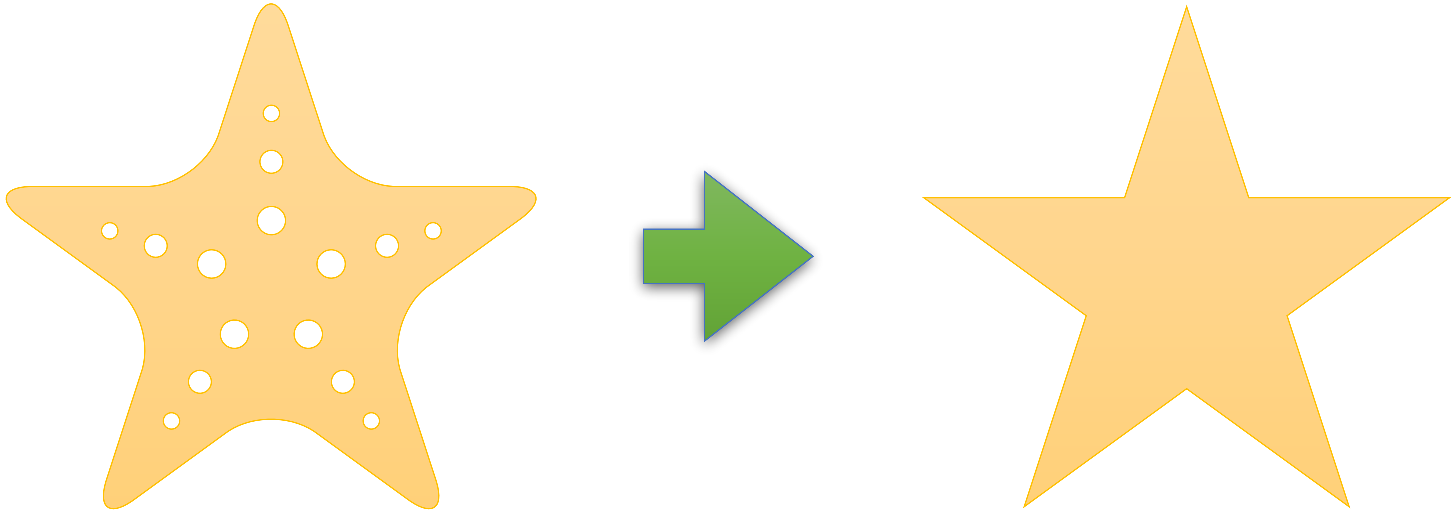
$$y = -8.9 + 4.6x_5 + 0.01x_6 + 12.0x_{15} + 17.5x_{21} + 0.2x_{22},$$

Picture 3

$$y = -76.7 + 9.3x_2 + 22.0x_7 - 13.2x_8 + 3.4x_{11} + 7.2x_{28}.$$

Breiman, Leo. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)." *Statistical science*16.3 (2001): 199-231.

Explainable Models



f - Original Model

g - Explanation Model

Explanation model, which we define as any interpretable approximation of the original model.

Definitions

- Given an input *x*, *f(x)* is a prediction given by *f*
- *x'* is a simplified input that map to the original input through some function x = $h_x(x')$
- Local methods try to ensure $g(z') \approx f(h_x(z'))$
- An additive feature attribution method have an explanation model that is a linear function of binary variables:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i,$$

- Where $z' \in \{0,1\}^M$, and M is the number of simplified input features, and $\phi \in \mathbb{R}$

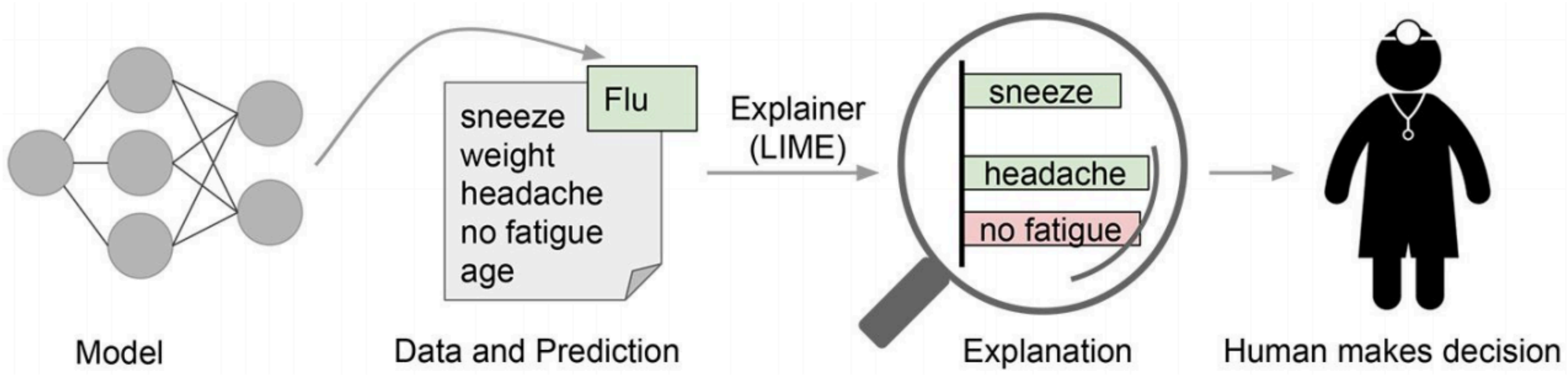
Summary

- Some new models:
  - LIME (2016)
  - DeepLIFT (2017)
  - Layer-Wise Relevance Propagation (2015)
  - SHAP (2017)

LIME

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 an obscure model.
- No need to know how it makes predictions

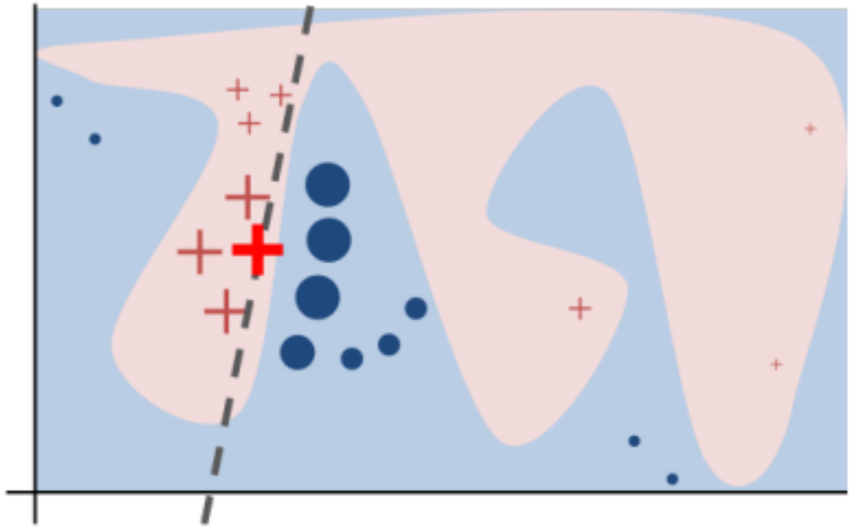


LIME: Output

- Blue variable values contribute to the classification of an instance
- Orange variable values are evidence against it

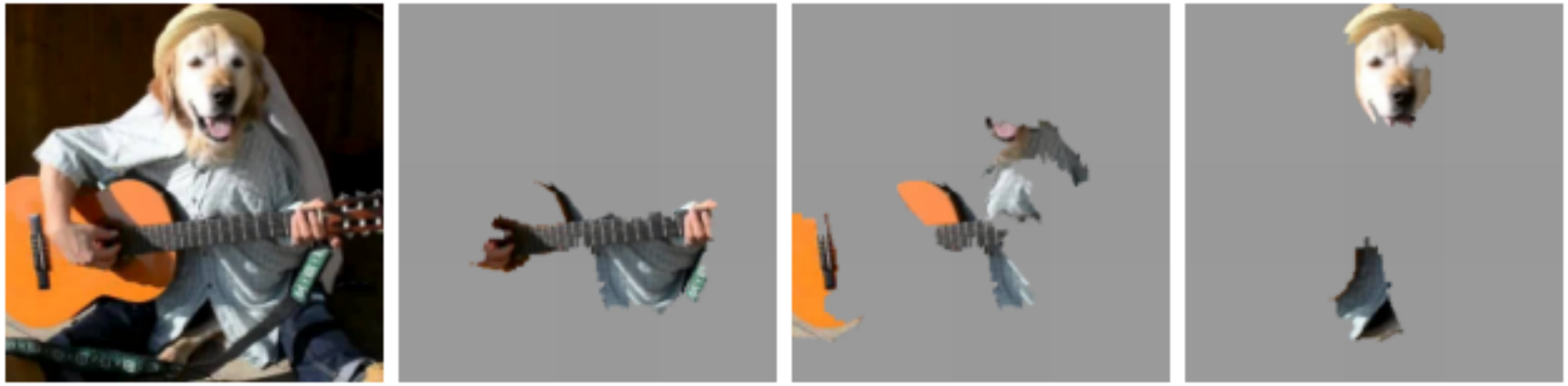
Process

1. Choose an observation to explain
2. Create new dataset around observation by sampling from distribution learnt on training data
3. Calculate distances between new points and observation, that's our measure of similarity
4. Use model to predict class of the new points
5. Find the subset of m features that has the strongest relationship with our target class
6. Fit a linear model on fake data in m dimensions weighted by similarity
7. Weights of linear model are used as explanation of decision



How LIME Works

- Simplified inputs x’ are considered interpretable inputs
- $x = h_x(x')$ converts a binary vector of interpretable inputs into the original input space
- For example, for images, h_x converts 1 to leaving a super pixel as its original value and 0 to replace the super pixel with an average of neighboring pixels (represents in being missing)



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Definitions

- Given an input *x*, *f(x)* is a prediction given by *f*
- *x'* is a simplified input that map to the original input through some function x = $h_x(x')$
- Local methods try to ensure $g(z') \approx f(h_x(z'))$
- An additive feature attribution method have an explanation model that is a linear function of binary variables:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i,$$

- Where $z' \in \{0,1\}M$, and M is the number of simplified input features, and $\phi \in \mathbb{R}$

Finding the right points

- To find ϕ, *LIME minimizes the following objective functions*

$$\xi = \underset{g \in \mathcal{G}}{\arg\min}\ L(f, g, \pi_{x'}) + \Omega(g).$$

- *Faithfulness of g(z') to the original model* $f(h_x(z'))$ is enforced through the locally weighted square loss function L over a set of samples in the simplified input space (weighted by the local kernel $\pi_{x'}$)
- *$\Omega(g)$ penalizes the complexity of g*

https://arxiv.org/pdf/1602.04938v1.pdf

Random Forest model

```
gs = GridSearchCV(rf_model, {"model__max_depth": [10, 15],
                             "model__min_samples_split": [5, 10]},
                  n_jobs=-1, cv=5, scoring="accuracy")

gs.fit(X_train, y_train)
```

```
In [42]:
accuracy_score(y_test, y_pred)
Out[42]:
0.8809581613660273
```

```
print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.94      0.92      0.93     10965
           1       0.48      0.57      0.52      1392

   micro avg       0.88      0.88      0.88     12357
   macro avg       0.71      0.75      0.73     12357
weighted avg       0.89      0.88      0.89     12357
```

Creating an explainer

```
explainer = LimeTabularExplainer(convert_to_lime_format(X_train,
categorical_names).values,
                                 mode="classification",
                                 feature_names=X_train.columns.tolist(),
                                 categorical_names=categorical_names,
                                 categorical_features=categorical_names.keys(),
                                 discretize_continuous=True,
                                 random_state=42)
```

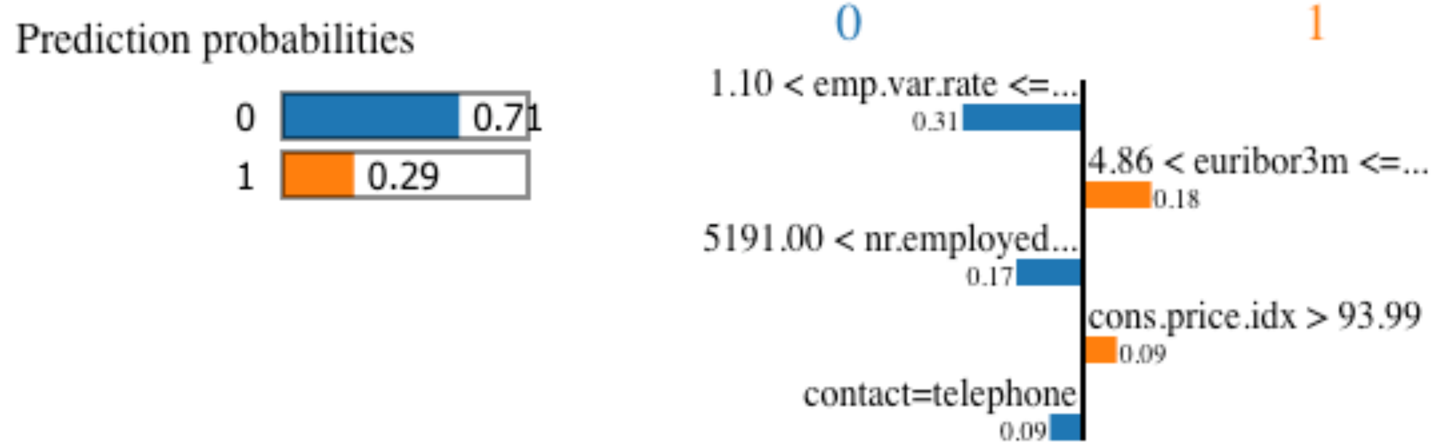
```
i = 2
X_observation = X_test.iloc[[i], :]
X_observation
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome
12077	35	technician	single	professional.course	no	no	no	telephone	jun	fri	397	1	999	0	nonexistent

https://github.com/klemag/pydata_nyc2018-intro-to-model-interpretability

Running the explainer

```
explanation = explainer.explain_instance(observation, lr_predict_proba, num_features=5)
explanation.show_in_notebook(show_table=True, show_all=False)
```



Feature	Value
emp.var.rate	1.40
euribor3m	4.95
nr.employed	5228.10
cons.price.idx	94.47
contact=telephone	True

Summary

- Linear approximation to localized models
- The inherent paradox of explaining models
- Depends on sampling of points, so it can be unstable