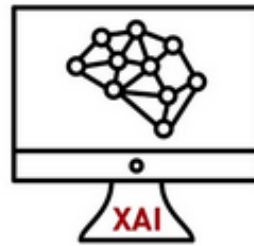# Session Agenda

Introduction

Importance of interpretable AI

LIME: What and How?

Hands on model interpretation

Final thoughts

About Me

Laisha Wadhwa

➢ Undergraduate at IIIT Sricity.

➢ Microsoft AI challenge 2018 winner

➢ Techgig Geek Goddess 2019 AIML hackathon winner

➢ Sabre Hack 2019 Winner

➢ Python and ML Enthusiast

➢ Working on autonomous driving

Icerties AIML Blockchain Hackathon 2019 Winner (2nd runner up)

We all have been here!
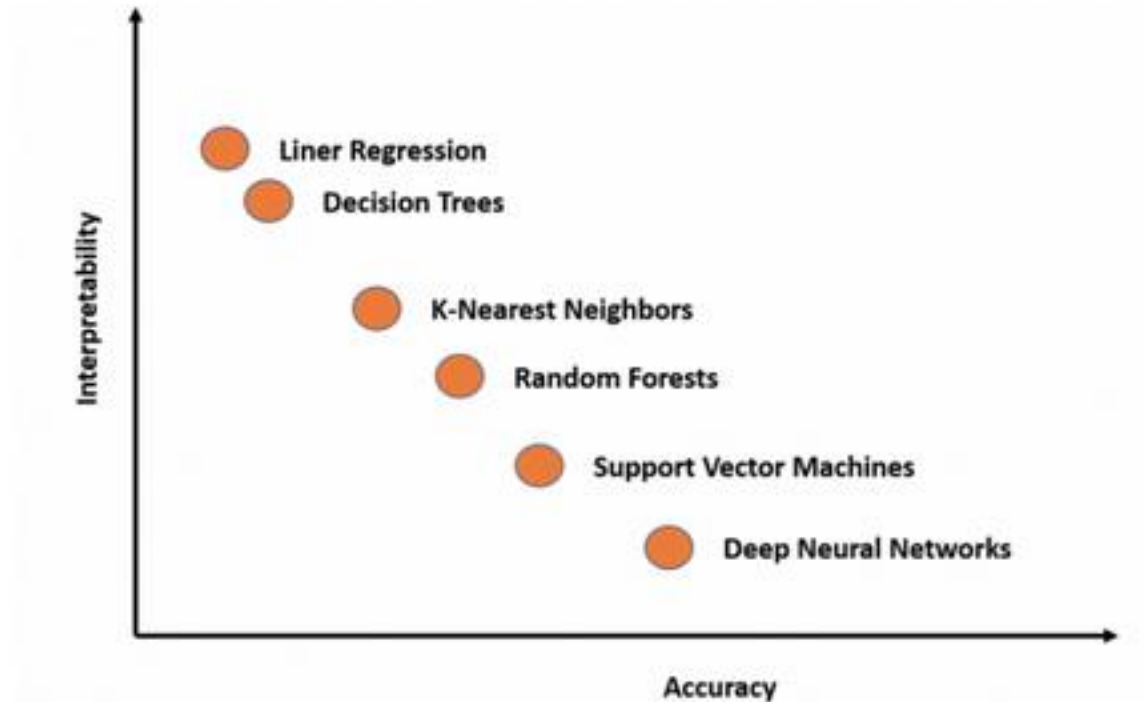
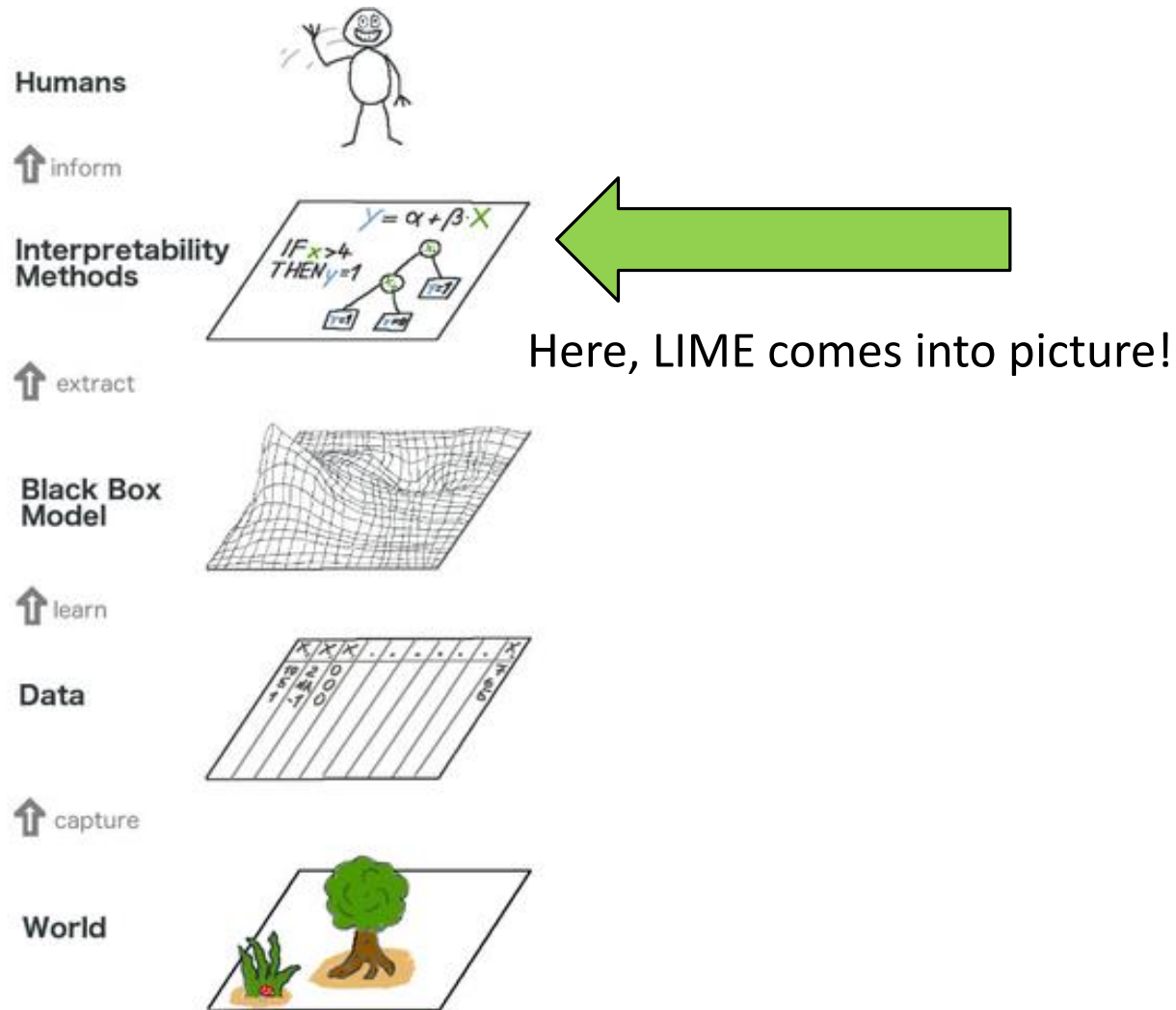**When your classifier achieves 98% accuracy**

Your classifier's Output
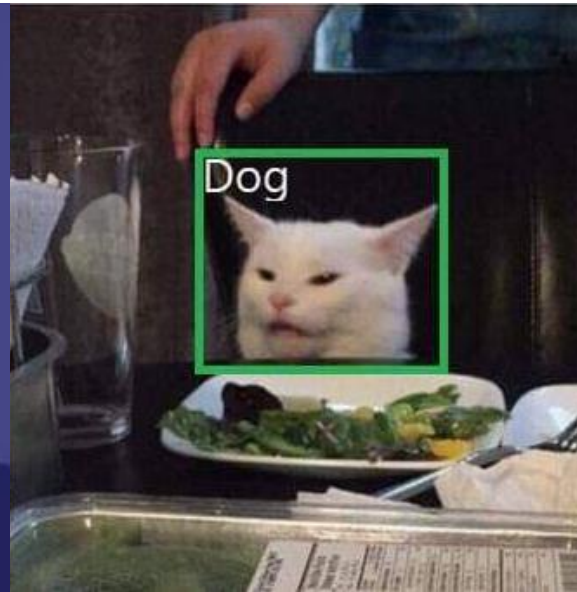
YOU

Accuracy vs Interpretability

ML and DL models have become a black-box

Humans

⬆ inform

Interpretability Methods

$$y = \alpha + \beta \cdot X$$

IF $x > 4$ THEN $y = 1$

Here, LIME comes into picture!

⬆ extract

Black Box Model

⬆ learn

Data

⬆ capture

World

The idea is to interpret the models and understand why the classifier chose a particular class or why the model gives more False Positives ??
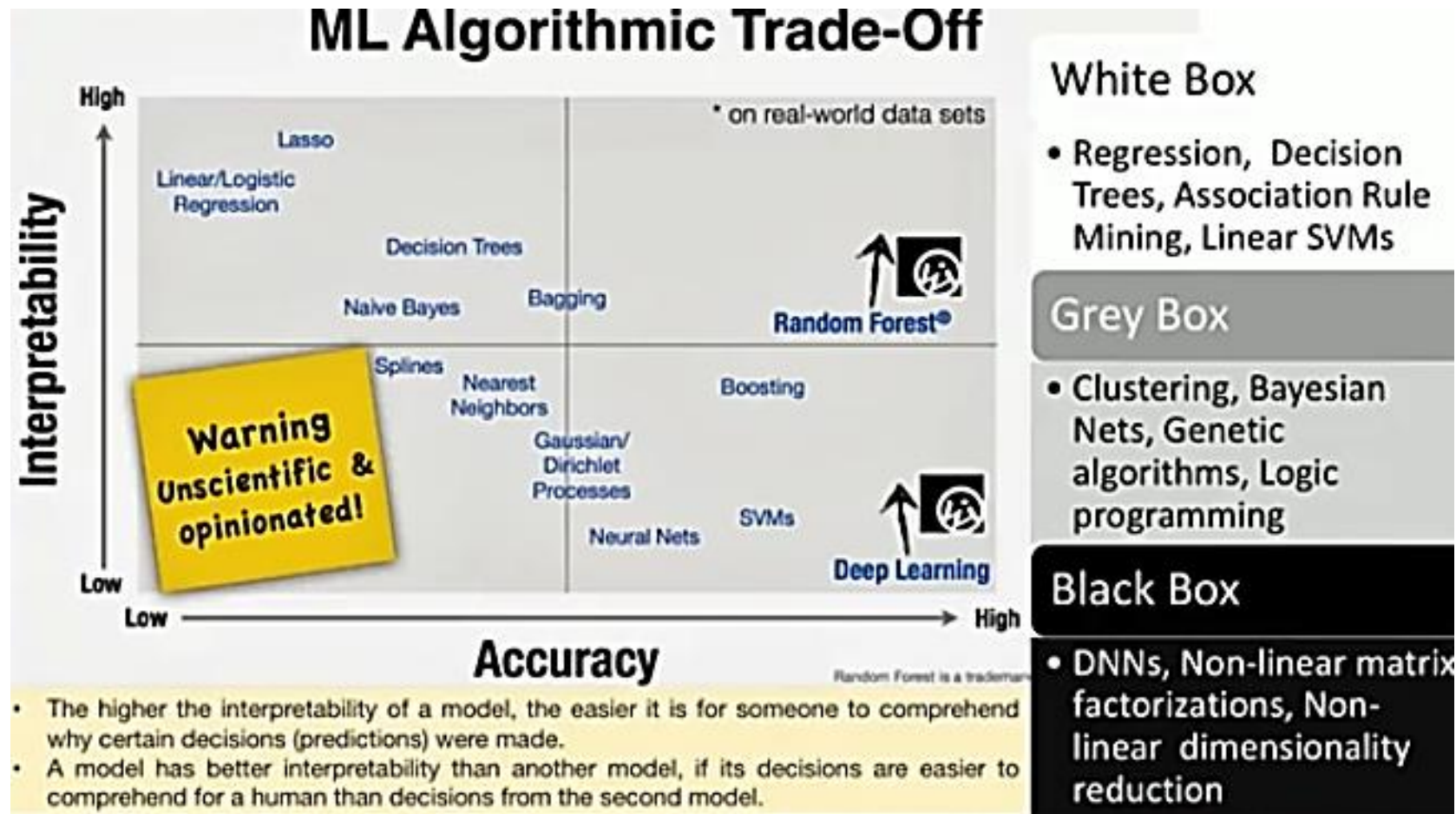
Slide credits: **Bring in the Lawyers: Explainable AI Driven Decision-making by Vikas Agrawal at ODSC_India**

What is LIME??

- LIME is a novel algorithm designed by Riberio Marco, Singh Sameer, Guestrin Carlos to assess the behaviour of the any model using local interpretable surrogate models and also an easy to use **python library.**

- LIME uses a representation that is understood by the humans irrespective of the actual features used by the model.

- LIME minimises the **locality-aware loss** which is measured using a mathematical formula (It's basic math!) which takes into consideration measure of **unfaithfulness of the model** in approximating the target variable and measure of model complexity (p) of explanation.

- E.g. if the model to be explained is decision tree, p can be depth of the tree or in case of linear explanation models it can be number of non zero weights.

Why should you use it?

LIME

> Interpretability.

> Explains decision boundaries of the model in human understandable form.

> Local Fidelity

> Model Agnostic

> Locally approximate global model

**Traditional Approach**

> Exploratory data Analysis

> Visualisation [t-SNE to visualize the famous MNIST hand-written digits]

> Model Performance Evaluation Metrics.

> Can't comprehend the decision boundary with changing nature of input points.

> Can't check how important features are in deciding model predictions and how well they might be working on new data points.
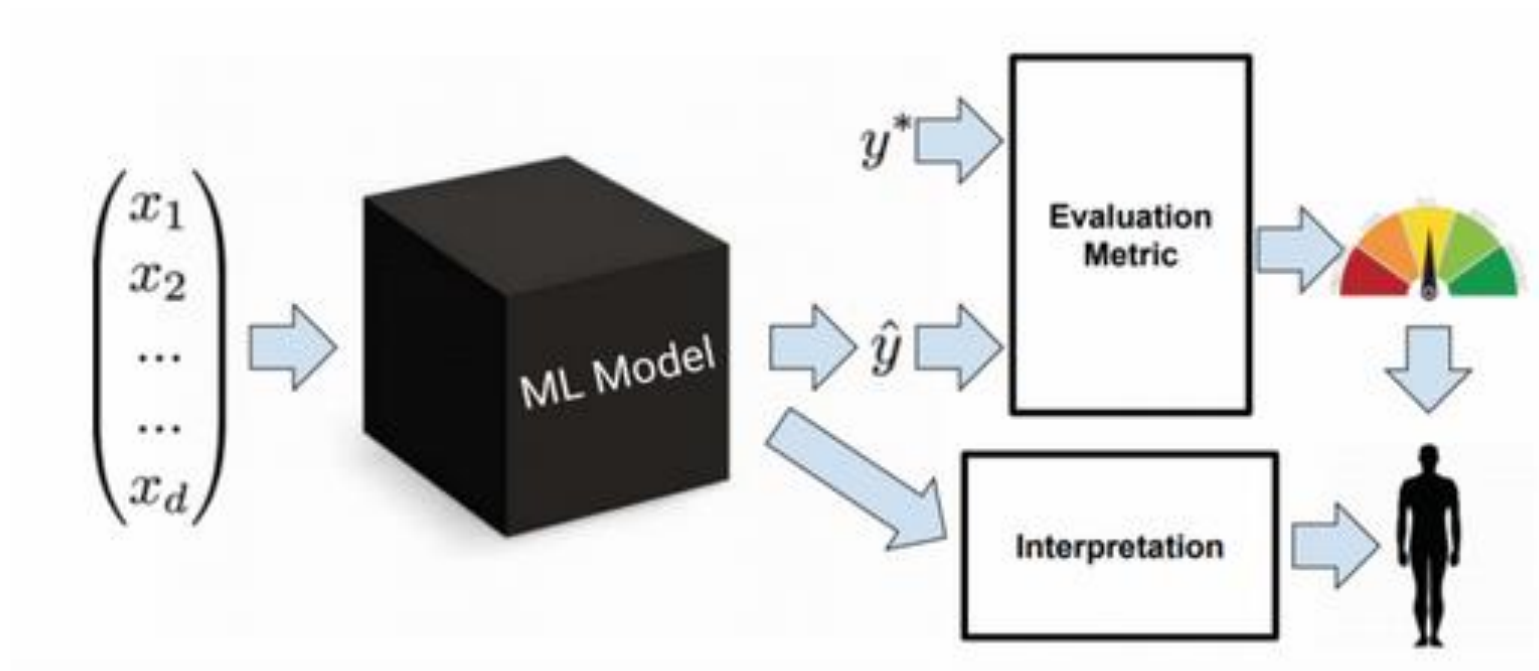
Traditional
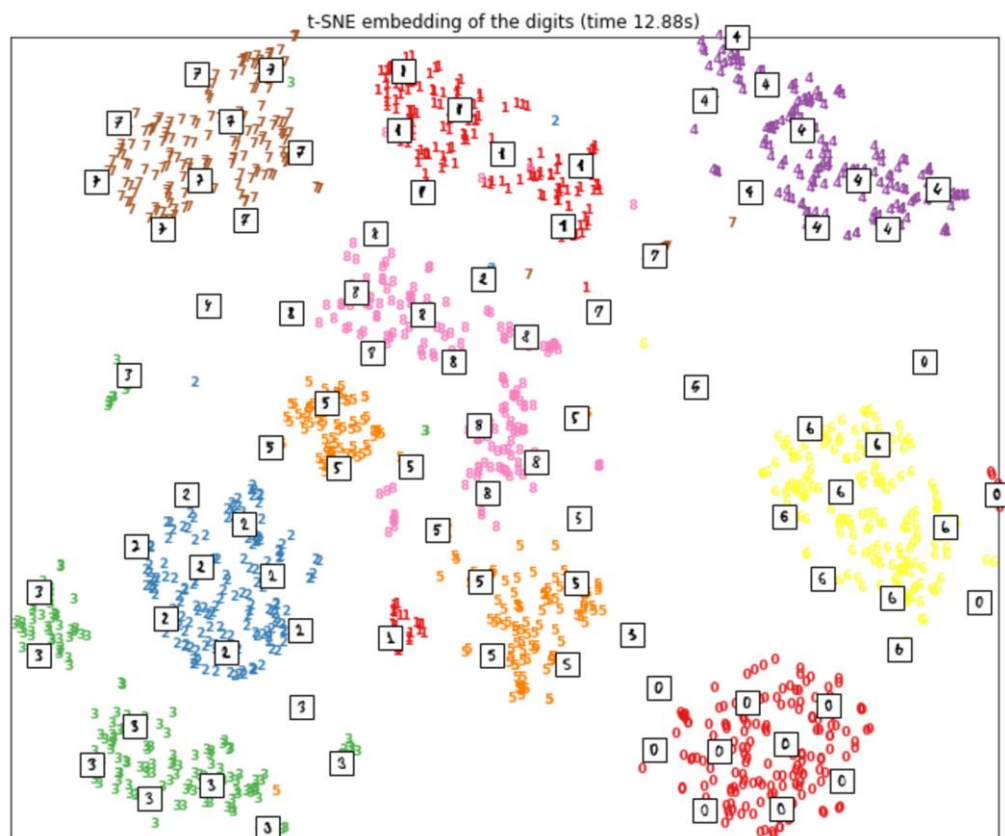
LIME
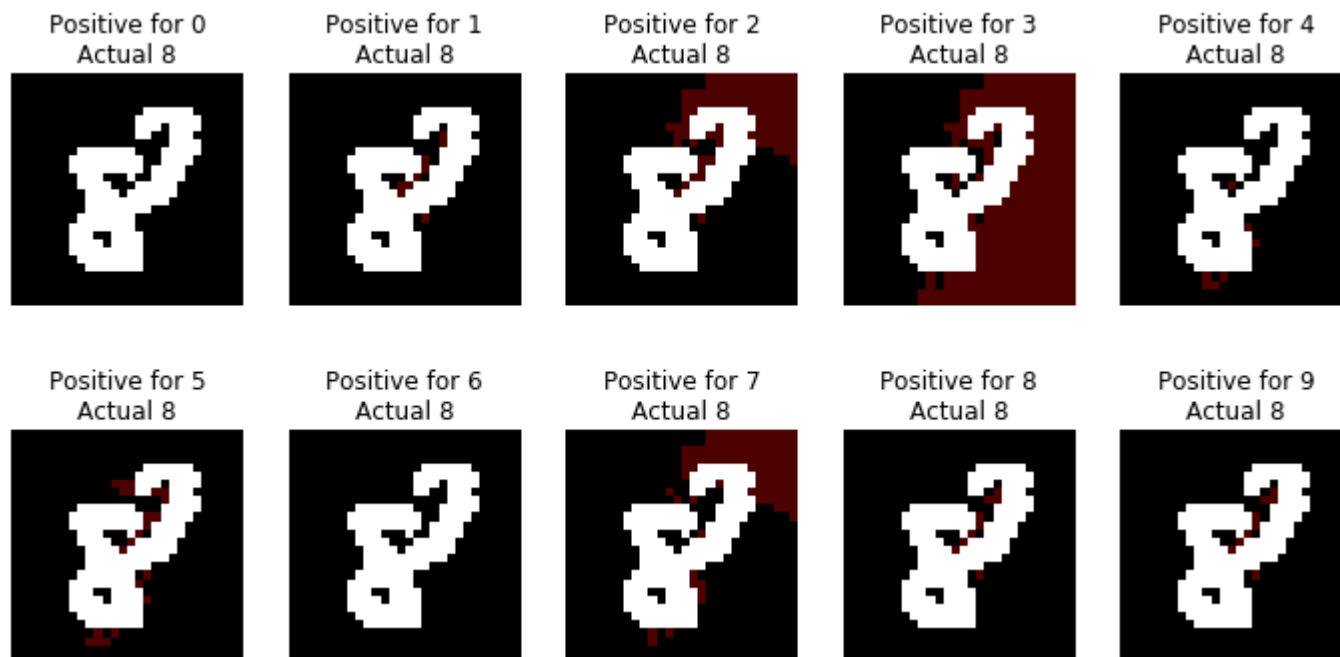
LIME helps to interpret models, HOW?

Using LIME to explain a complex model's prediction for input x_i

1. Sample points around x_i
2. Use complex model to predict labels for each sample
3. Weigh samples according to distance to x_i
4. Learn new simple model on weighted samples
5. Use simple model to explain

Ack: Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin

Slide credits: **Bring in the Lawyers: Explainable AI Driven Decision-making by Vikas Agrawal at ODSC_India**
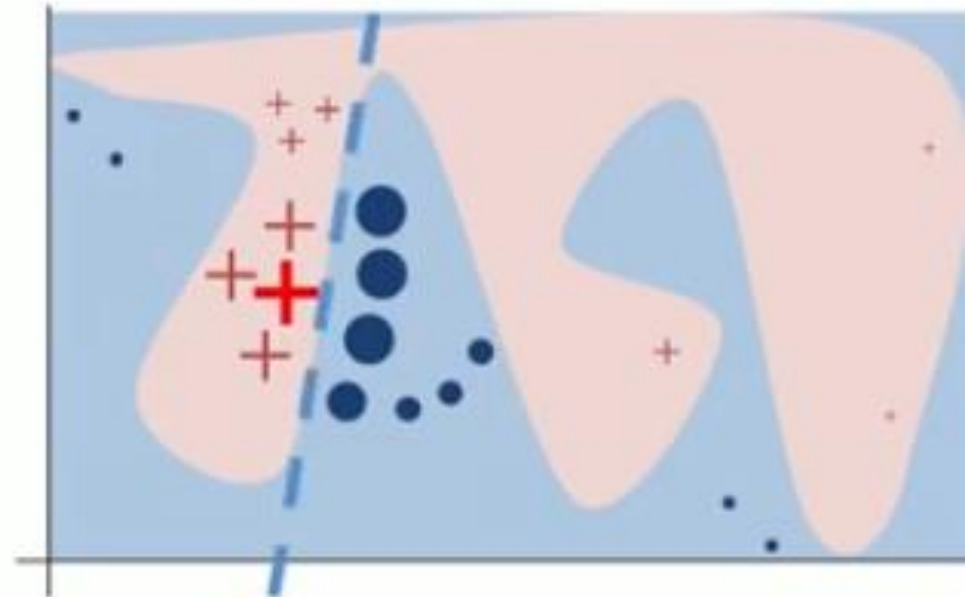
The Math – *locality-aware loss*

$$\xi(x) = \operatorname*{argmin}_{g \in G} \; \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

First Term : measure of unfaithfulness of g in approximating f in the locality defined by Pi. This is termed as **locality-aware loss** in the original paper

Last term : measure of model complexity of explanation g. For example if your explanation model is decision tree it can be depth of the tree or in case of linear explanation models it can be number of non zero weights

So what all can LIME do?

❖It can explain models with any kind of data:

a) **Text** : It represents presence/absence of words. LimeTextExplainer

b) **Image** : It represents presence/absence of super pixels (contiguous patch of similar pixels). LimeImageExplainer

c) **Tabular data** : It is a weighted combination of columns. LimeTabularExplainer

DIGIT classification

Data & Labels

Network training

0
1
2
3
4
5
6
7
8
9
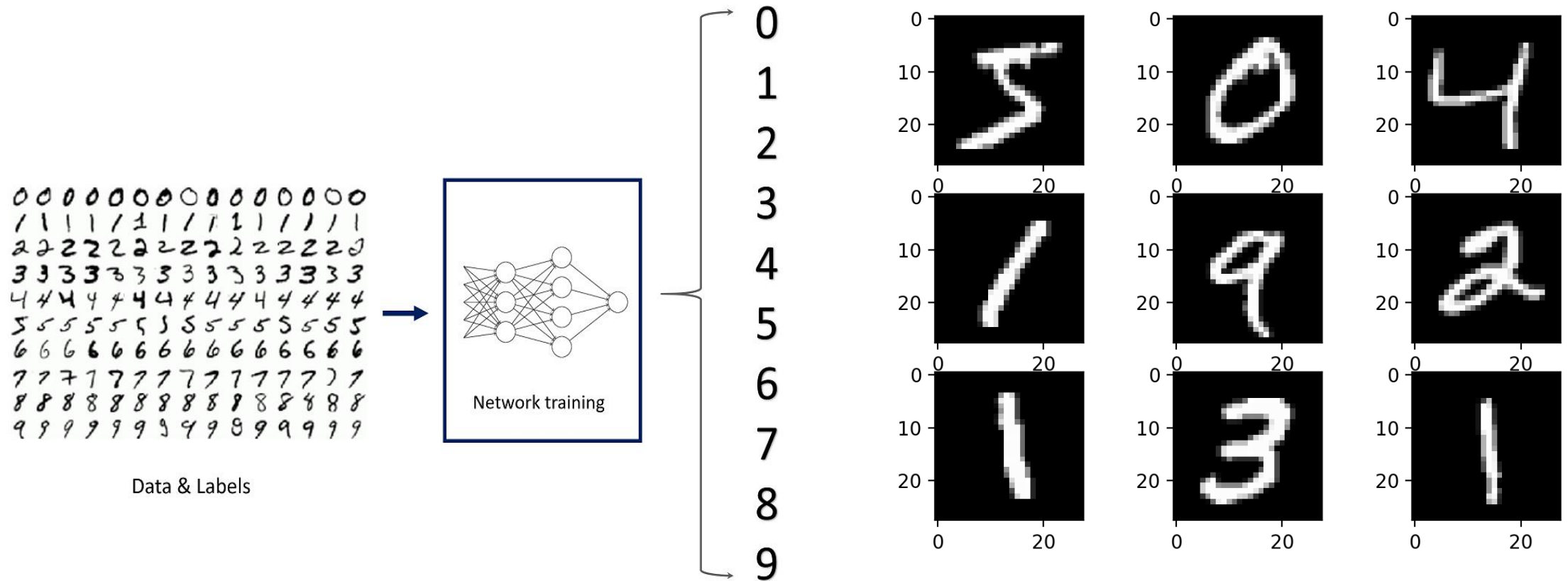
How do we do it??

```
[65]  #Setting up the main pipeline
      simple_dt_pipeline = Pipeline([
          ('Make Gray', makegray_step),
          ('Flatten Image', flatten_step),
          #('Normalize', Normalizer()),
          #('PCA', PCA(16)),
          ('DT', DecisionTreeClassifier(criterion = "gini",
                  random_state = 100,max_depth=7, min_samples_leaf=5) )
                                  ])
```

LIME: explain_instance and ImageExplainer

```
explainer = lime_image.LimeImageExplainer(verbose = False)
segmenter = SegmentationAlgorithm('quickshift', kernel_size=1, max_dist=200, ratio=0.2)
```

```
#testidx can be changed to check for different examples
testidx = 7
explanation = explainer.explain_instance(X_test[testidx],
                              classifier_fn = simple_dt_pipeline.predict_proba,
                              top_labels=10, hide_color=0, num_samples=10000, segmentation_fn=segmenter)
```

Let's see it in Action!

- LIME tests out what happens to your black box model's predictions when you feed variations or perturbations of your dataset.

- How? By generating a new set comprising of perturbed samples and the corresponding black box model's predictions.

- LIME then trains an interpretable model weighted by the proximity of the sampled instances to the instance of interest.

Let's see how well can a decision tree differentiate the numbers!!

Let's delve deeper

Let's say we are trying to classify wolves and dogs.

Wolf

Dog

What if we have wolves and dogs in entirely different backgrounds? This is entirely possible as wolves are mostly found in the wild (in snow, jungles, etc.) while dogs are in completely different backgrounds (households) generally.

We built an image classifier and got a really good performance on the validation set. But when we test the classifier we get false positives? Can't seem to figure out why?

LIME has the answer!

We observe that, our model is actually **learning the background pretty well!!**

Let's see this in action! [Cat dog classification exercise]

Image classification- How to?

Loading a model:

```
#Loading model
checkpoints_dir = '/content/tf-models/slim/pretrained'
init_fn = slim.assign_from_checkpoint_fn(
    os.path.join(checkpoints_dir, 'inception_v3.ckpt'),
    slim.get_model_variables('InceptionV3'))
init_fn(session)
```

Running inference on a given image

```
def predict_fn(images):
    return session.run(probabilities, feed_dict={processed_images: images})
```

```
preds = predict_fn(images)
for x in preds.argsort()[0][-5:]:
    print (x, names[x], preds[0,x])
```

Which LIME class to use?

We use LIME.ImageExplainer

explainer = lime_image.LimeImageExplainer()

Using the explainer object we run inference on a single image and set the parameters like top_labels etc

explanation = explainer.explain_instance(image, predict_fn, top_labels=10, hide_color=0, num_samples=1000)

Interpret and Improve

❖ So, LIME can help interpret visual outputs.

❖ Can it help us understand text models and term importance? [text word importance hands on]

❖

We have seen digits, cats and dogs..

Let's see Text classification

Well be using Stack overflow question and tag data.

	post	tags
0	what is causing this behavior in our c# datet...	c#
1	have dynamic html load as if it was in an ifra...	asp.net
2	how to convert a float value in to min:sec i ...	objective-c
3	.net framework 4 redistributable just wonderi...	.net
4	trying to calculate and print the mean and its...	python

How will we do it?

- We'll first convert words to vector using CountVectorizer()

```
Vectorizer = CountVectorizer(analyser='word', token_pattern=r'\w(1,)', ngram_range=(1, 3), stop_words = 'english', binary=True)
train_vectors = vectorizer.fit_transform(X_train)
```

Next , we build a Logistic regression model for text data

```python
#iniatializing the model
logreg = LogisticRegression(n_jobs=1, C=1e5)

#fitting the model with train_vectors
logreg.fit(train_vectors, y_train)
```

For LIME we use: LimeTextExplainer which takes class_names as a parameter
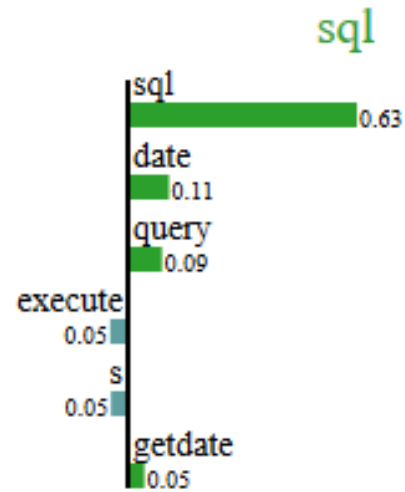
```python
class_names=list(df.tags.unique())



#initialising an explainer for LIME
explainer = LimeTextExplainer(class_names=class_names)
```
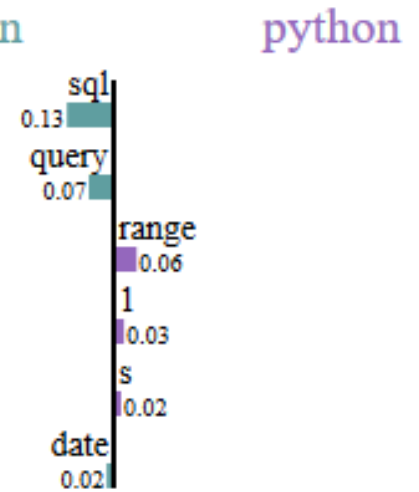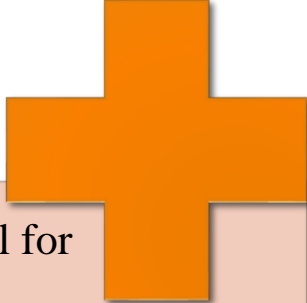
Let's check how it works

Things to Explore

You can check out similar library for interpretability called: SHAP
(https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d)

A Guide for Making Black Box Models Explainable by **Christoph Molnar** is a great book to refer to if you want to delve deep and make machine learning more interpretable.

Conclusion

1. Sampling from a Gaussian distribution.

2. Complexity of the explanation model has to be defined in advance.

1. Local, interpretable model for explanation.

2. Explanations are short, selective and possibly contrastive.

3. Works for tabular data, text and images.

4. **Fidelity Measure:** A reliable measure of interpretability.

5. Can use other (interpretable) features than the original model was trained on.

What have we learnt?

➢ Don't trust models without explanations !

➢ Interpreting the machine's intelligence is tough (For Humans as well).

➢ For Business centric use cases we must ensure that model is learning the scene description well irrespective of the accuracy. (Dog wolf example!).

The Future of Interpretability

✓ The focus will be on model-agnostic interpretability tools.

✓ Machine learning will be automated and, with it, interpretability.

✓ We do not analyze data, we analyze models.

> Test whether A or B is better: For this we can also use partial dependence functions. What we do not have yet are statistical tests for arbitrary black box models.

✓ The data scientists will automate themselves.

✓ Interpretability could boost machine intelligence research.

References

[1] "Why Should I Trust You?" Explaining the Predictions of Any Classifier — Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin.

[2] Blog by the authors, https://homes.cs.washington.edu/~marcotcr/blog/lime/

[3] Github link : https://github.com/marcotcr/lime

[4] Link to original paper : https://arxiv.org/abs/1602.04938