Explainable Artificial Intelligence

Introduction

- AI is critical to many sectors in Information Technologies

Introduction

- AI is critical to many sectors in Information Technologies
- Today’s AI is achieving unprecedented levels of performance
- Becoming increasingly important

Better Go/Chess Player

More “Realistic” Text Generation

ChatGPT

Introduction

- AI is critical to many sectors in Information Technologies
- Today’s AI is achieving unprecedented levels of performance
- Becoming increasingly important

Better Go/Chess Player

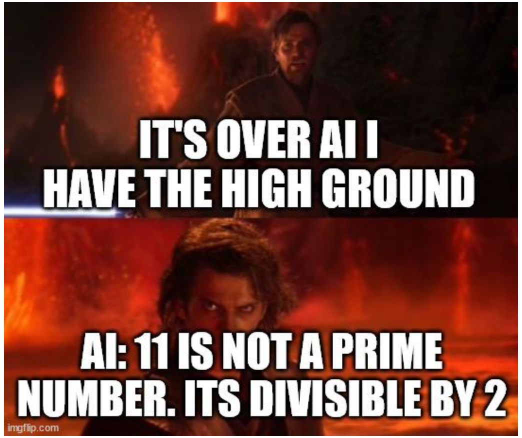
More “Realistic” Text Generation

ChatGPT

Equal to Humans at Drawing Hands

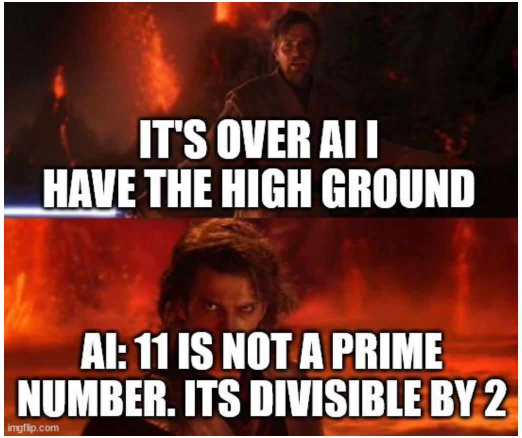
Unfortunately

- Deployment in critical fields faces an uphill battle



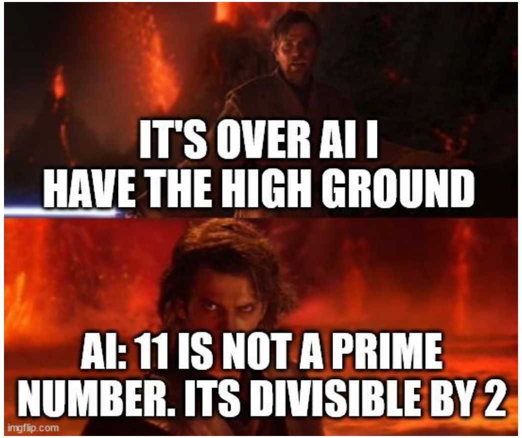
Unfortunately

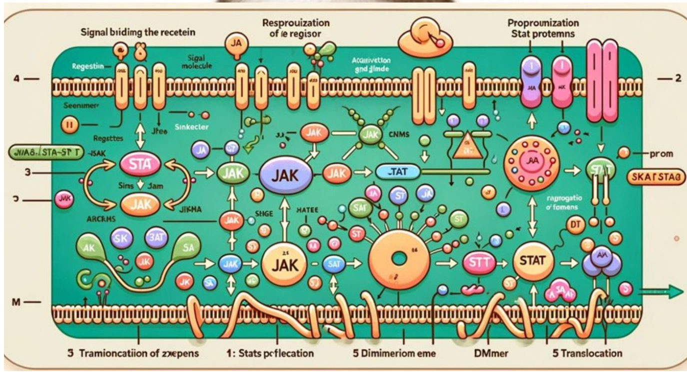
- Deployment in critical fields faces an uphill battle
- Decisions that affect humans can be difficult to accept



Unfortunately

- Deployment in critical fields faces an uphill battle
- Decisions that affect humans can be difficult to accept
- Just like how “journals” shouldn’t be accepting generative ai created papers





Unfortunately

- Deployment in critical fields faces an uphill battle
- Decisions that affect humans can be difficult to accept
- Just like how “journals” shouldn’t be accepting generative ai created papers

RETRACTED: Cellular functions of spermatogonial stem cells in relation to JAK/STAT signaling pathway Retracted

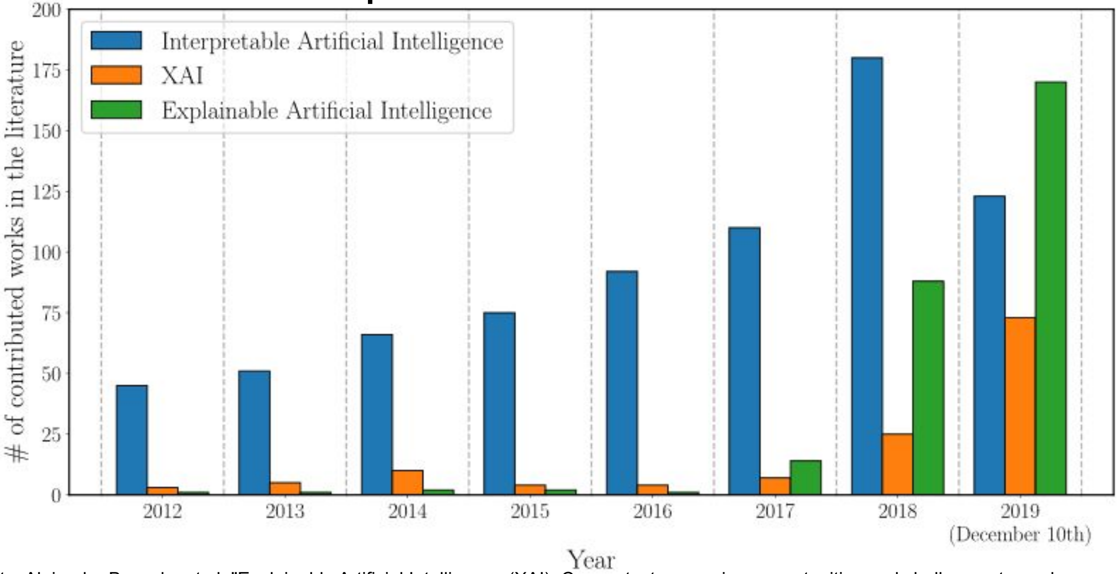
Thus

- There is need in understanding decisions made by AI

Thus

- There is need in understanding decisions made by AI
- The first AI systems were interpretable
- The current age of AI is dominated by opaque decision processes

Explainable Research Growth

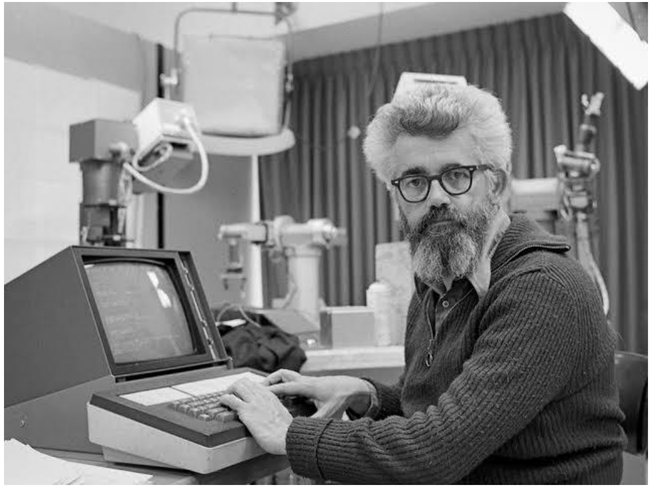


Arrieta, Alejandro Barredo, et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." *Information fusion* 58 (2020): 82-115.

History Lesson

- AI dates back to the 1950's

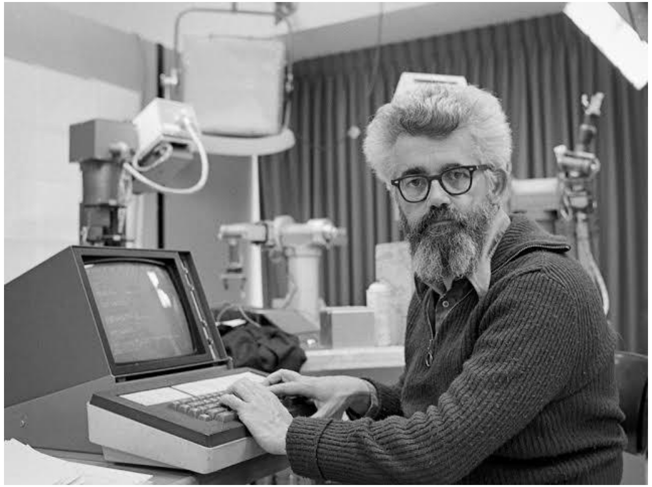
John McCarthy



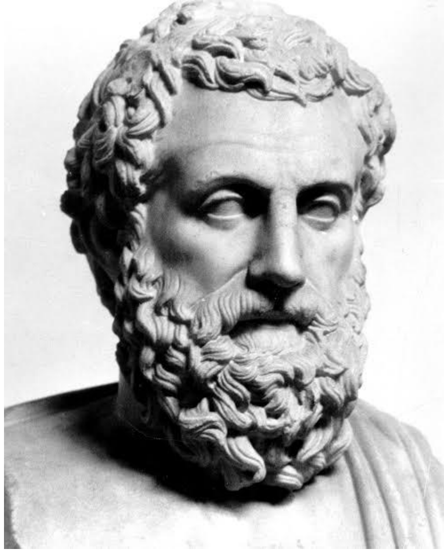
History Lesson

- AI dates back to the 1950's
- Even further if you consider Aristotle and Greek myths

John McCarthy



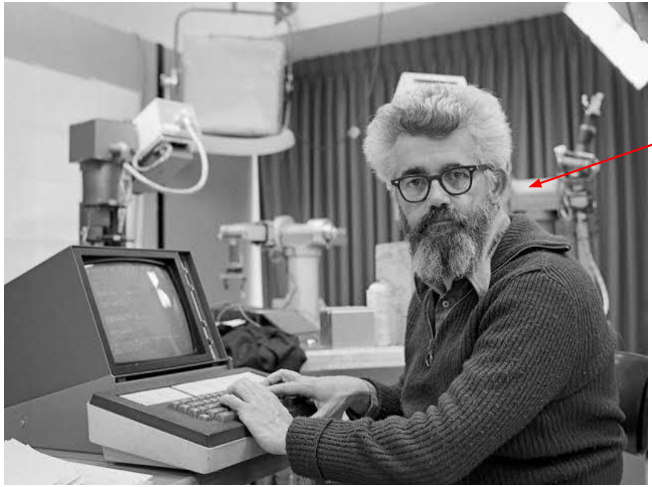
Aristotle



History Lesson

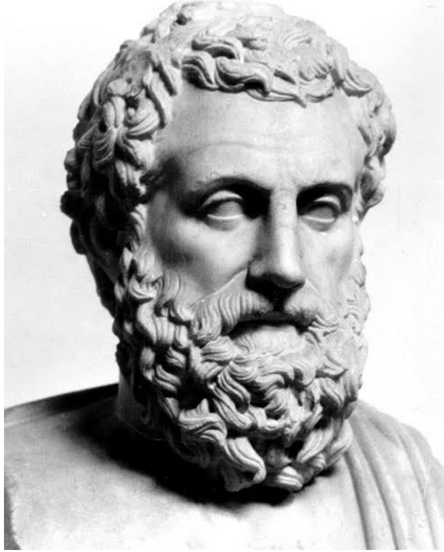
- AI dates back to the 1950's
- Even further if you consider Aristotle and Greek myths

John McCarthy



Old school cool

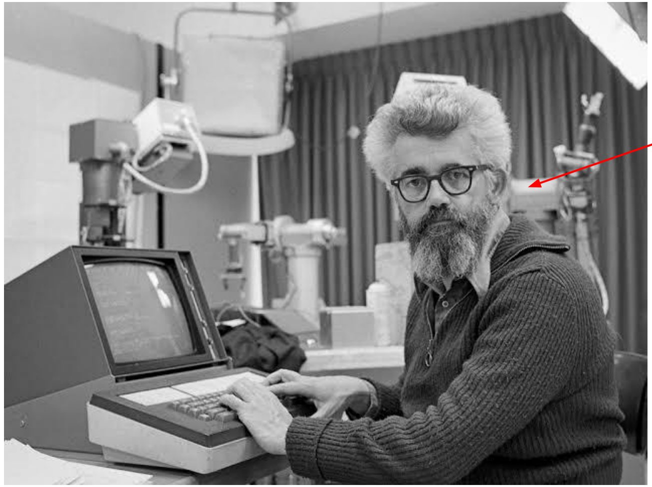
Aristotle



History Lesson

- AI dates back to the 1950's
- Even further if you consider Aristotle and Greek myths

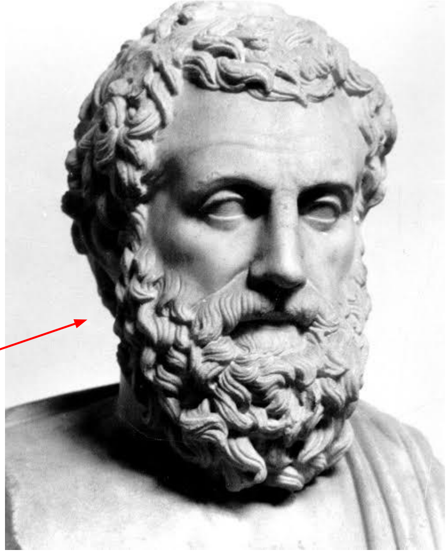
John McCarthy



Old school cool

Literally immortalized into stone

Aristotle



History Lesson

- The "Golden Age" of AI 1956-1974
- Micro-Worlds by Marvin Minsky

"The question of whether a computer can think is no more interesting than the question of whether a submarine can swim."

— Edsger Dijkstra

History Lesson

- The “Golden Age” of AI 1956-1974
- Micro-Worlds by Marvin Minsky
  - Simplify the world into a very specific set of information

“The question of whether a computer can think is no more interesting than the question of whether a submarine can swim.”

— Edsger Dijkstra

History Lesson

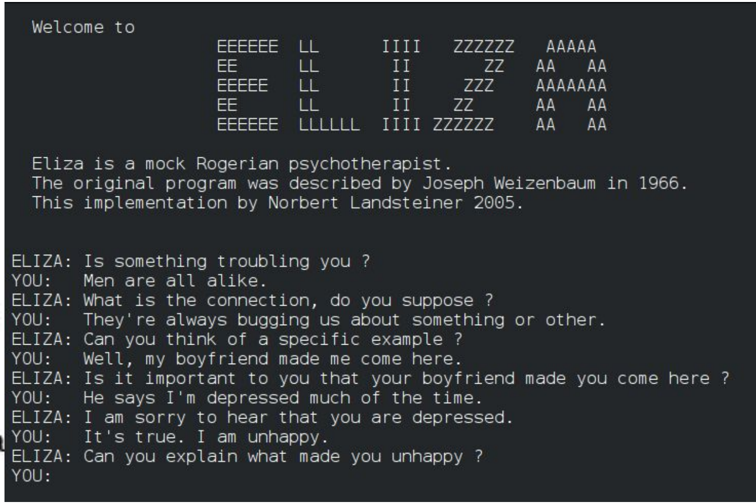
- The “Golden Age” of AI 1956-1974
- Micro-Worlds by Marvin Minsky
  - Simplify the world into a very specific set of information
- John McCarthy’s development of Lisp

“The question of whether a computer can think is no more interesting than the question of whether a submarine can swim.”

— Edsger Dijkstra

History Lesson

- The “Golden Age” of AI 1956-1974
- Micro-Worlds by Marvin Minsky
  - Simplify the world into a very specific set of information
- John McCarthy’s development of Lisp
- The first chatbot ELIZA
- Lots of research funded by DARPA



“The question of whether a computer can think is no more interesting than the question of whether a submarine can swim.”

— Edsger Dijkstra

History Lesson

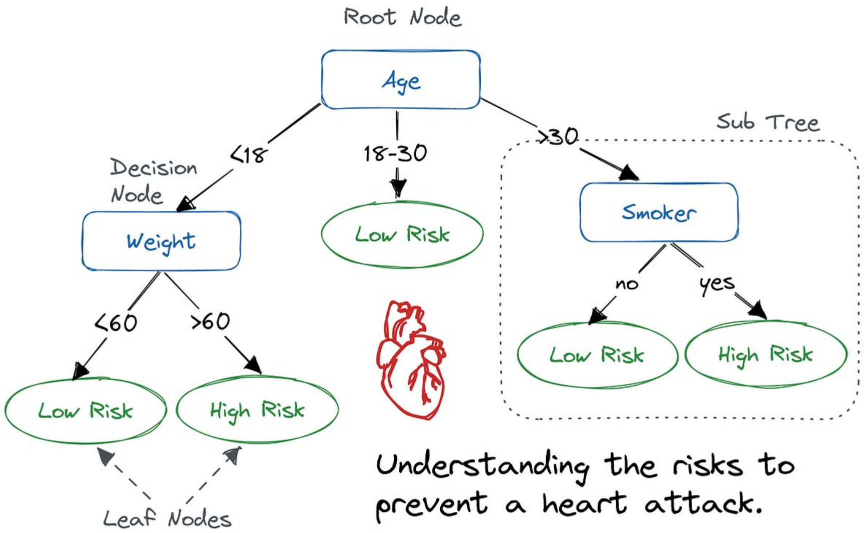
- Early AI was considered “white-box”

History Lesson

- Early AI was considered “white-box”
- What were some examples of “white-box” AI algorithms?

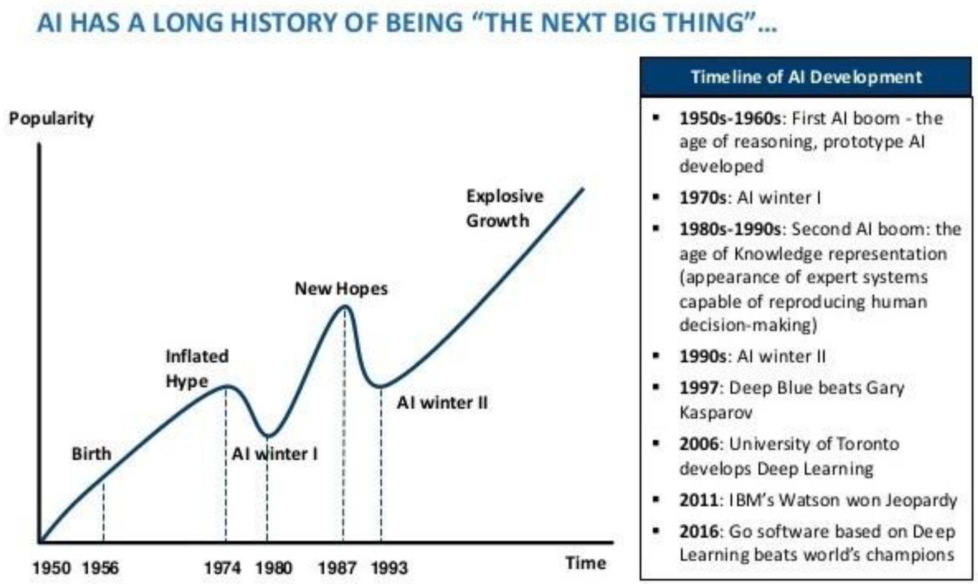
History Lesson

- Early AI was considered “white-box”
- Rule-based algorithms
- Linear decision
- Clustering



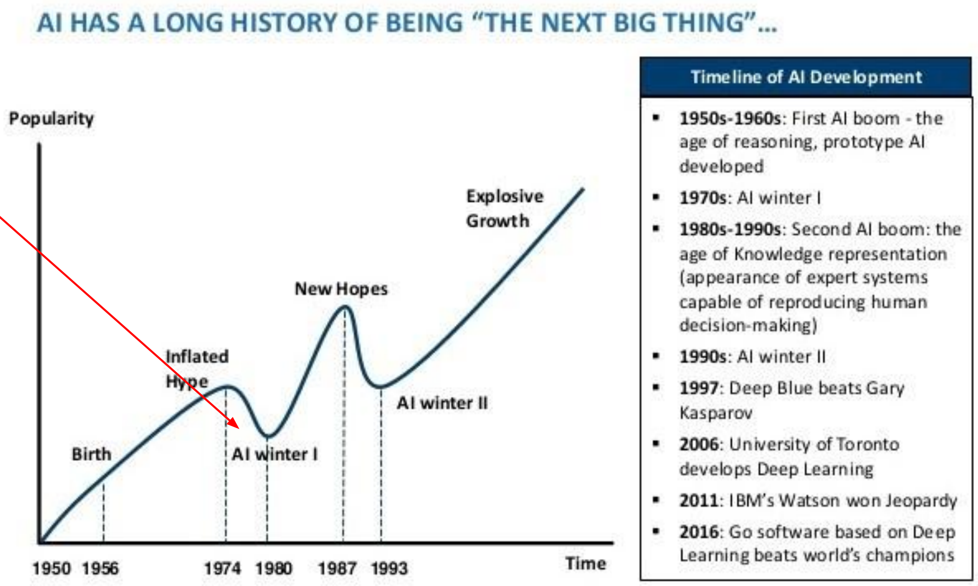
History Lesson

- The first AI winter in the 1970s



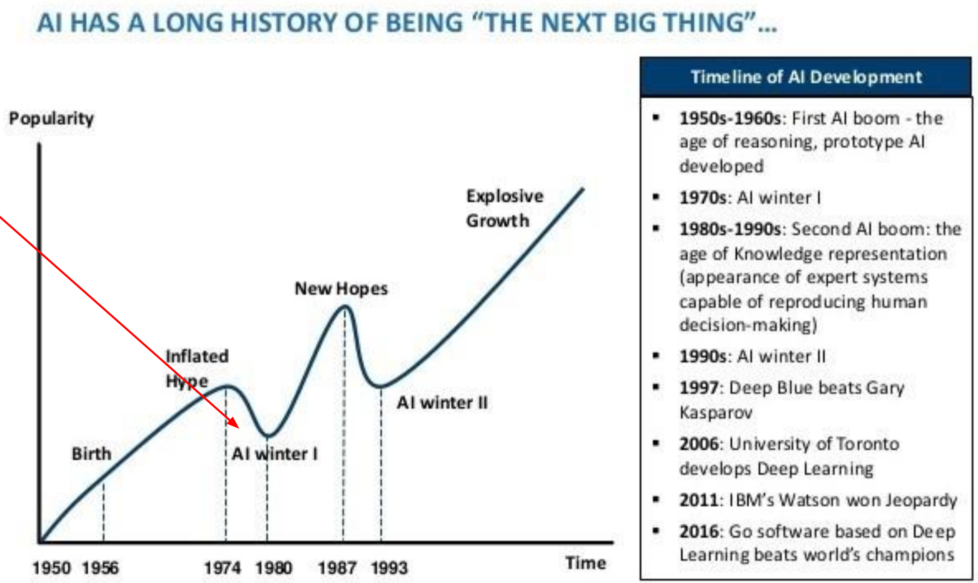
History Lesson

- The first AI winter in the 1970s
- Funding dried up, popularity diminished



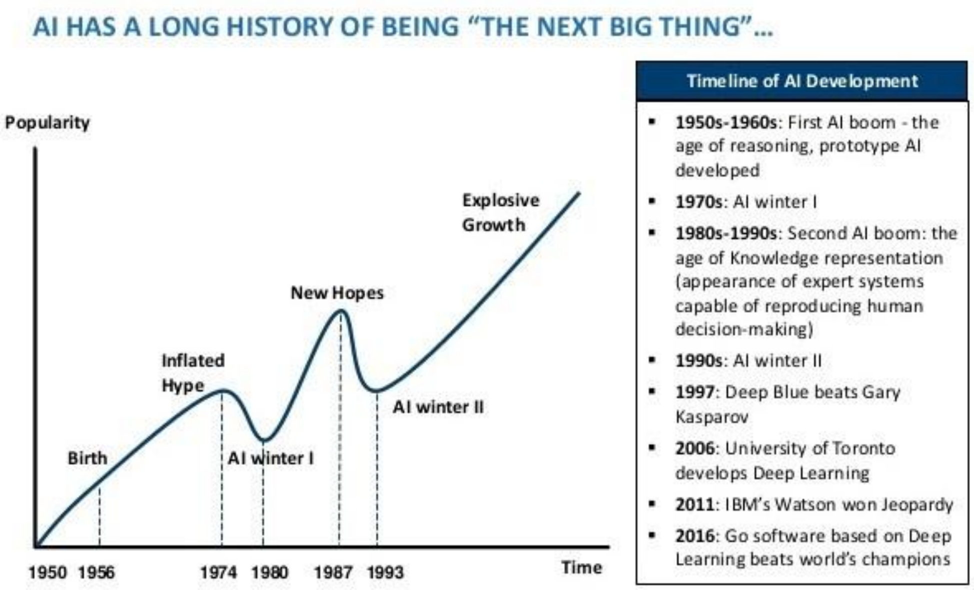
History Lesson

- The first AI winter in the 1970s
- Funding dried up, popularity diminished
- Why?



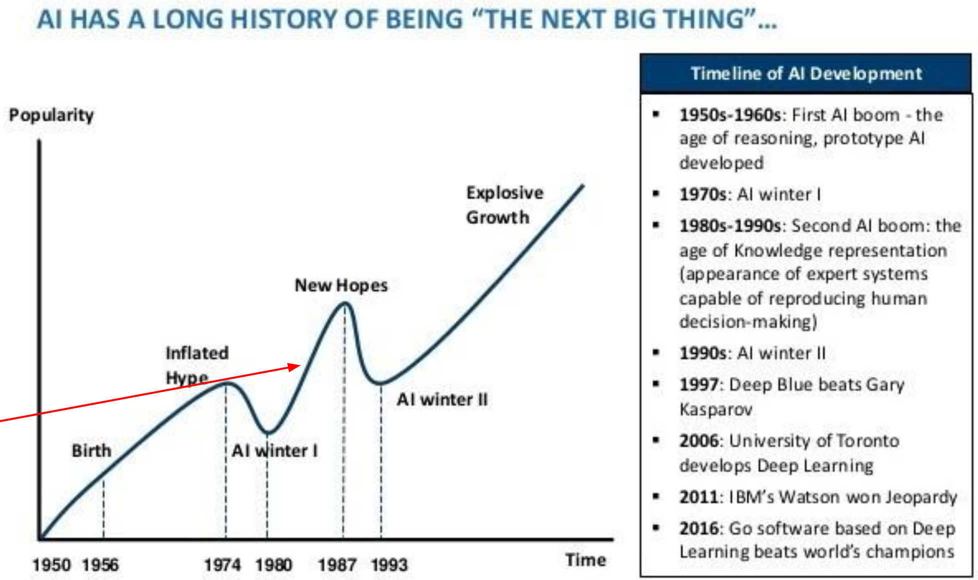
History Lesson

- The first AI winter in the 1970s
- Funding dried up, popularity diminished
- Likely
  - Technical limitations
  - Unrealistic expectations
  - No practical use at the time



History Lesson

- The first AI winter in the 1970s
- Funding dried up, popularity diminished
- Likely
  - Technical limitations
  - Unrealistic expectations
  - No practical use at the time
- Rise in expert systems

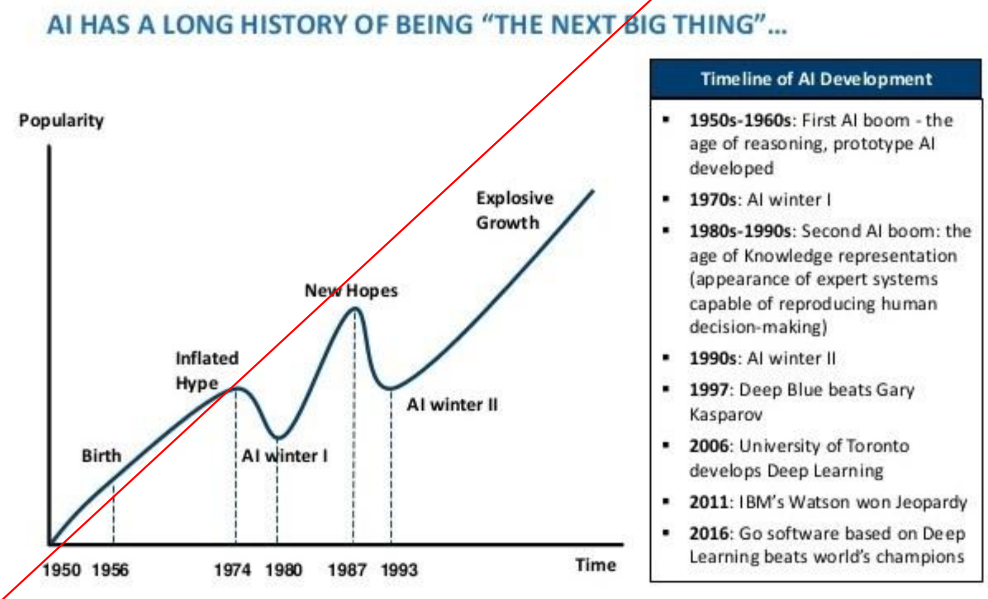


History Lesson

- The first AI winter in the 1970s
- Funding dried up, popularity diminished
- Likely
  - Technical limitations
  - Unrealistic expectations
  - No practical use at the time
- Rise in expert systems
- Fall because of technical limits

History Lesson

- The first AI winter in the 1970s
- Funding dried up, popularity diminished
- Likely
  - Technical limitations
  - Unrealistic expectations
  - No practical use at the time
- Rise in expert systems
- Fall because of technical limits
- Moore’s law and Neural networks



Historical XAI

- Dates back to 1970’s

Historical XAI

- Dates back to 1970’s
- MYCIN for medical diagnostic systems
- DENDRAL for chemical analysis
- Both used rule-based explanations
- Also “canned” XAI systems

Historical XAI

- Dates back to 1970’s
- MYCIN for medical diagnostic systems
- DENDRAL for chemical analysis
- Both used rule-based explanations
- Also “canned” XAI systems
  - Pre generated responses about domain specific decisions

Back to Today

- Deep Neural Networks are used everywhere
- They are highly accurate
- But they are not trustworthy, tractable, or interpretable

Back to Today

- Deep Neural Networks are used everywhere
- They are highly accurate
- But they are not trustworthy, tractable, or interpretable

- Increase in demand for “**ethical AI**”
- Need to understand trade off of **transparency** to **performance**
- Systems can be **improved** by **understanding** them

Back to Today

- Deep Neural Networks are used everywhere
- They are highly accurate
- But they are not trustworthy, tractable, or interpretable

- Increase in demand for “**ethical AI**”
- Need to understand trade off of **transparency** to **performance**
- Systems can be **improved** by **understanding** them
  - This is the idea that researchers ignore/miss

Important Terminology

- **Understandability** – characteristic of a model to make a human understand its function (how the model works) without the need for explaining its internal structure or algorithmic processes

Arrieta, Alejandro Barredo, et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." *Information fusion* 58 (2020): 82-115.

Important Terminology

- **Understandability** – characteristic of a model to make a human understand its function (how the model works) without the need for explaining its internal structure or algorithmic processes
- **Comprehensibility** – ability of a learned model to represent its learned knowledge to a human

Arrieta, Alejandro Barredo, et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." *Information fusion* 58 (2020): 82-115.

Important Terminology

- **Understandability** – characteristic of a model to make a human understand its function (how the model works) without the need for explaining its internal structure or algorithmic processes
- **Comprehensibility** – ability of a learned model to represent its learned knowledge to a human
- **Interpretability** – ability to explain or to provide meaning in understandable terms to a human

Arrieta, Alejandro Barredo, et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." *Information fusion* 58 (2020): 82-115.

Important Terminology

- **Understandability** – characteristic of a model to make a human understand its function (how the model works) without the need for explaining its internal structure or algorithmic processes
- **Comprehensibility** – ability of a learned model to represent its learned knowledge to a human
- **Interpretability** – ability to explain or to provide meaning in understandable terms to a human
- **Explainability** – Notion of explanations as an interface between humans and AI (Reasoning, Strengths and Weaknesses, Future behavior)

Arrieta, Alejandro Barredo, et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." *Information fusion* 58 (2020): 82-115.

Important Terminology

- **Understandability** – characteristic of a model to make a human understand its function (how the model works) without the need for explaining its internal structure or algorithmic processes
- **Comprehensibility** – ability of a learned model to represent its learned knowledge to a human
- **Interpretability** – ability to explain or to provide meaning in understandable terms to a human
- **Explainability** – Notion of explanations as an interface between humans and AI (Reasoning, Strengths and Weaknesses, Future behavior)
- **Transparency** – A model is considered to be transparent if by itself it is understandable

Arrieta, Alejandro Barredo, et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." *Information fusion* 58 (2020): 82-115.

Important Terminology

- **Understandability** – characteristic of a model to make a human understand its function (how the model works) without the need for explaining its internal structure or algorithmic processes
    - The authors believe that **Understandability** is the most crucial concept in XAI

Arrieta, Alejandro Barredo, et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." *Information fusion* 58 (2020): 82-115.

Important Terminology

- **Understandability** – characteristic of a model to make a human understand its function (how the model works) without the need for explaining its internal structure or algorithmic processes
  - The authors believe that **Understandability** is the most crucial concept in XAI
- **Explainability** – Notion of explanations as an interface between humans and AI (Reasoning, Strengths and Weaknesses, Future behavior)
  Transparency – A model is considered to be transparent if by itself it is understandable
  - I believe that the usability and transpareness of an AI system is the most crucial concept in XAI
  - Understandability is also important

Arrieta, Alejandro Barredo, et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." *Information fusion* 58 (2020): 82-115.

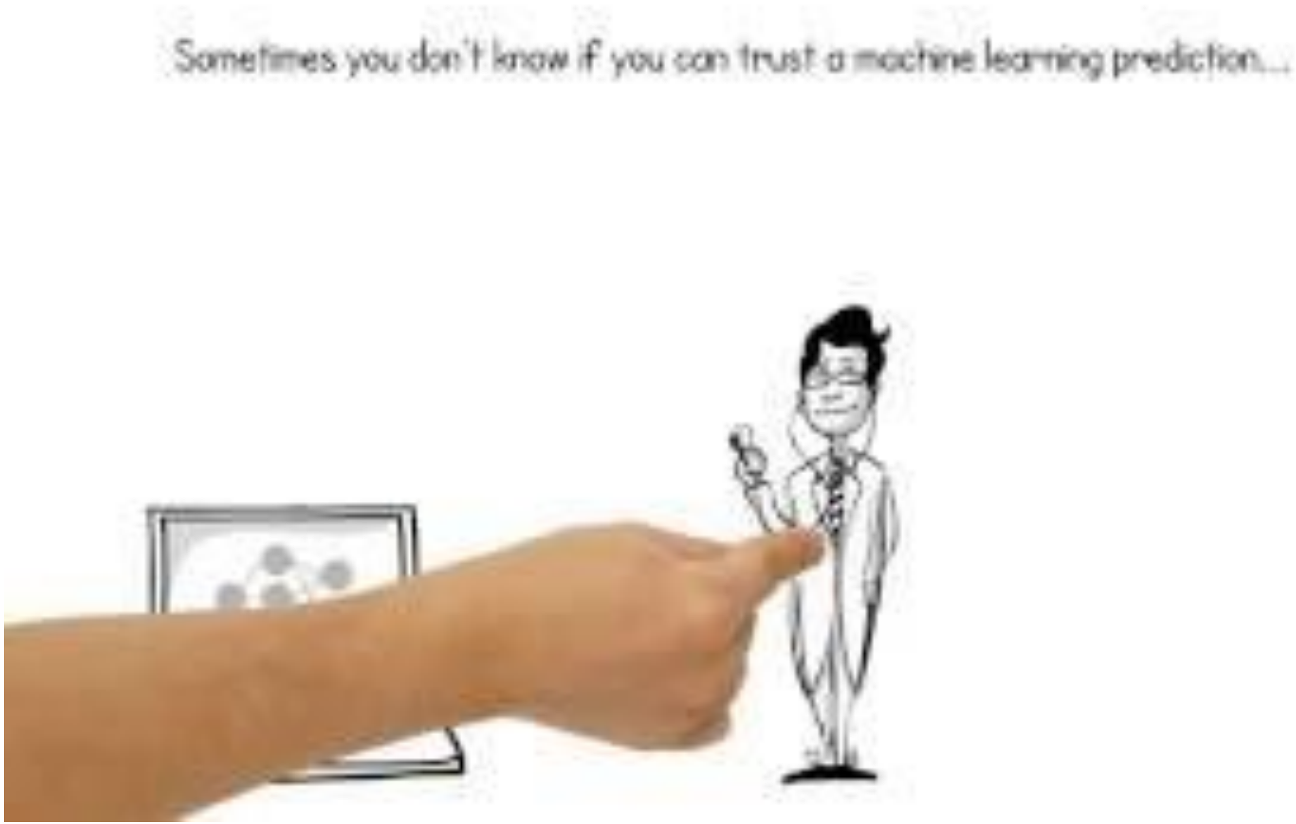
Other Important Terminology

- **Local Explanation** – interpretation that provides insights into a specific instance or data point

Other Important Terminology

- **Local Explanation** – interpretation that provides insights into a specific instance or data point
- **Global Explanation** – provides understanding of a model's overall behavior across the entire dataset

Local Interpretable Model-agnostic Explanations (LIME)



Local Interpretable Model-agnostic Explanations (LIME)

- LIME creates local explanations
- It follows this general set of steps to create explanations

Local Interpretable Model-agnostic Explanations (LIME)

- LIME creates local explanations
- It follows this general set of steps to create explanations
  - Pick a specific data point that you want explained

Local Interpretable Model-agnostic Explanations (LIME)

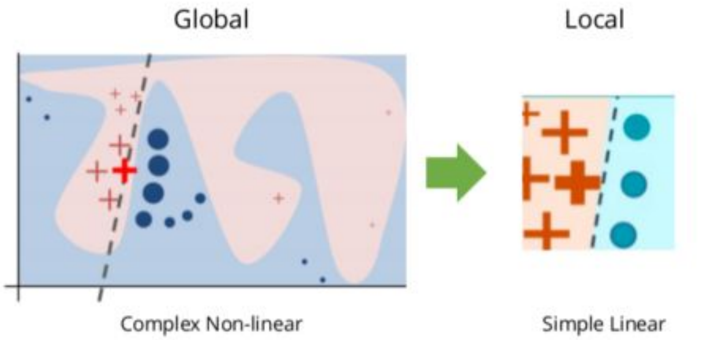
- LIME creates local explanations
- It follows this general set of steps to create explanations
  - Pick a specific data point that you want explained
  - LIME creates a **perturbed** dataset of that sample

Local Interpretable Model-agnostic Explanations (LIME)

- LIME creates local explanations
- It follows this general set of steps to create explanations
  - Pick a specific data point that you want explained
  - LIME creates a **perturbed** dataset of that sample
  - Your black box model is used to make predictions on the **perturbed** dataset

Local Interpretable Model-agnostic Explanations (LIME)

- LIME creates local explanations
- It follows this general set of steps to create explanations
  - Pick a specific data point that you want explained
  - LIME creates a **perturbed** dataset of that sample
  - Your black box model is used to make predictions on the **perturbed** dataset
  - LIME trains a surrogate model based on the model predictions

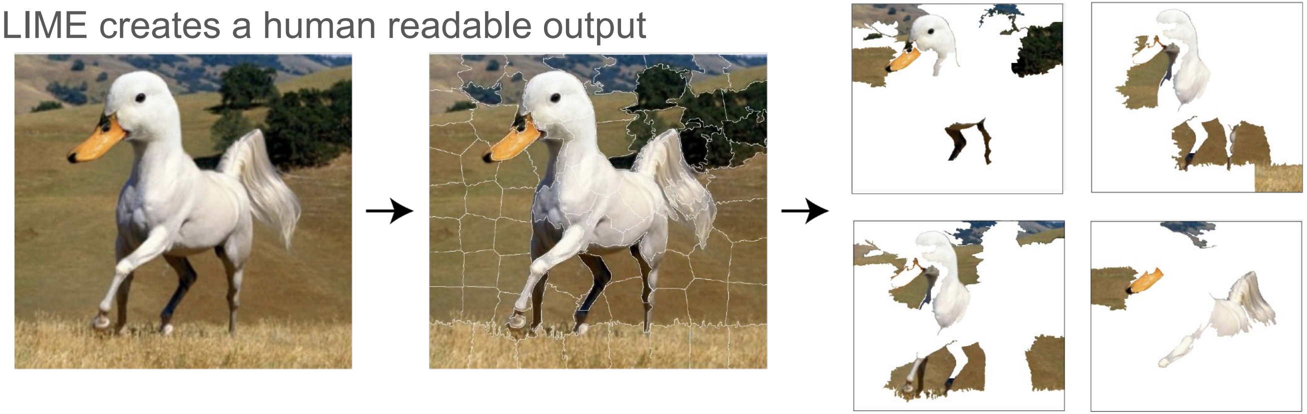


Local Interpretable Model-agnostic Explanations (LIME)

- LIME creates local explanations
- It follows this general set of steps to create explanations
  - Pick a specific data point that you want explained
  - LIME creates a **perturbed** dataset of that sample
  - Your black box model is used to make predictions on the **perturbed** dataset
  - LIME trains a surrogate model based on the model predictions
  - Surrogate is datamined to determine local feature importance

Local Interpretable Model-agnostic Explanations (LIME)

- LIME creates local explanations
- It follows this general set of steps to create explanations
  - Pick a specific data point that you want explained
  - LIME creates a **perturbed** dataset of that sample
  - Your black box model is used to make predictions on the **perturbed** dataset
  - LIME trains a surrogate model based on the model predictions
  - Surrogate is datamined to determine local feature importance
  - LIME creates a human readable output



Label: standard poodle
Probability: 0.18
Explanation Fit: 0.37

Label: goose
Probability: 0.15
Explanation Fit: 0.55

SHapley Additive exPlanations (SHAP)

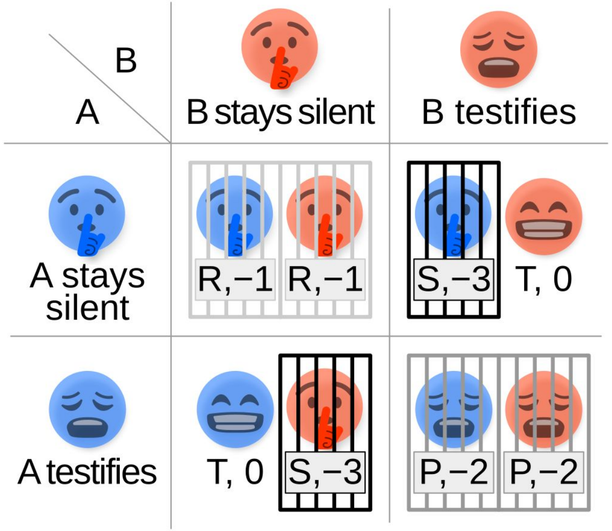
- Based off of cooperative game-theory

SHapley Additive exPlanations (SHAP)

- Based off of cooperative game-theory
    - **Players** – features in a model
    - **Coalitions** – feature subsets
    - **Payoff** – model predictions for an instance

SHapley Additive exPlanations (SHAP)

- Based off of cooperative game-theory
  - **Players** – features in a model
  - **Coalitions** – feature subsets
  - **Payoff** – model predictions for an instance
- Famous example of cooperative game-theory
- Prisoner's Dilemma
  - Two rational agents
  - Can either cooperate or stay silent
  - Dilemma is that the payoff is higher if the agents cooperate



SHapley Additive exPlanations (SHAP)

- SHAP can create both local and global explanations

SHapley Additive exPlanations (SHAP)

- SHAP can create both local and global explanations
- It follows this general set of steps to create explanations

SHapley Additive exPlanations (SHAP)

- SHAP can create both local and global explanations
- It follows this general set of steps to create explanations
  - Uses the model to make a baseline prediction (without features)

SHapley Additive exPlanations (SHAP)

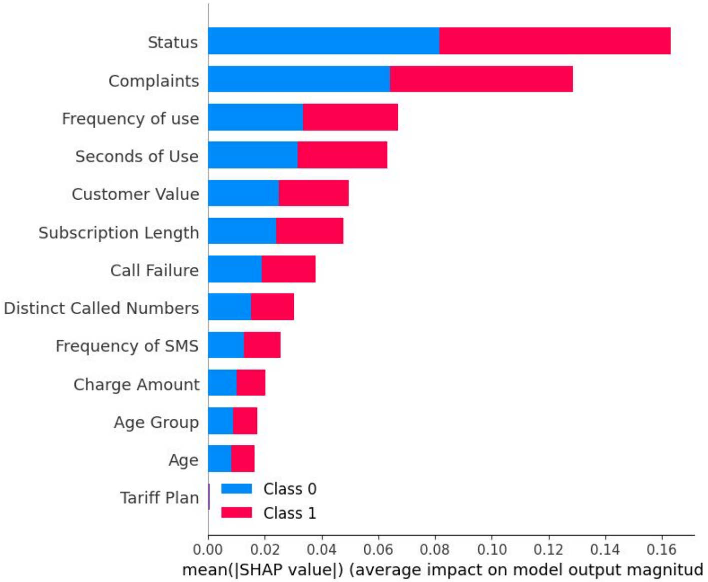
- SHAP can create both local and global explanations
- It follows this general set of steps to create explanations
  - Uses the model to make a baseline prediction (without features)
  - For every feature
    - Consider all subsets (coalitions) that exclude that feature
    - Add that feature to each coalition and calculate the change in model predictions

SHapley Additive exPlanations (SHAP)

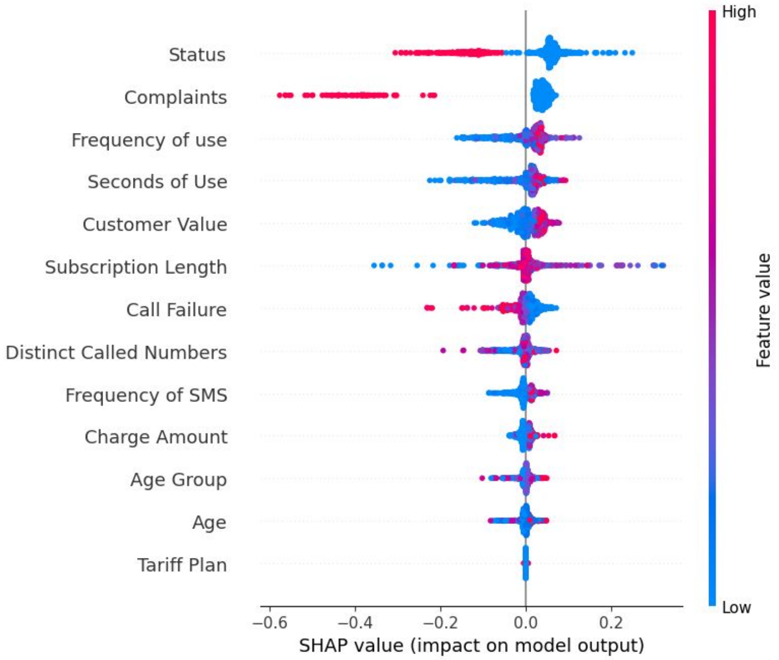
- SHAP can create both local and global explanations
- It follows this general set of steps to create explanations
    - Uses the model to make a baseline prediction (without features)
    - For every feature
        - Consider all subsets (coalitions) that exclude that feature
        - Add that feature to each coalition and calculate the change in model predictions
    - Compute the marginal contributions of the selected feature

SHapley Additive exPlanations (SHAP)

- SHAP can create both local and global explanations
- It follows this general set of steps to create explanations
  - Uses the model to make a baseline prediction (without features)
  - For every feature
    - Consider all subsets (coalitions) that exclude that feature
    - Add that feature to each coalition and calculate the change in model predictions
  - Compute the marginal contributions of the selected feature
  - Using the marginal contributions, calculate the additive shapley value

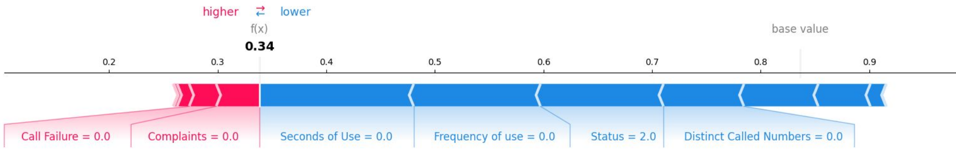


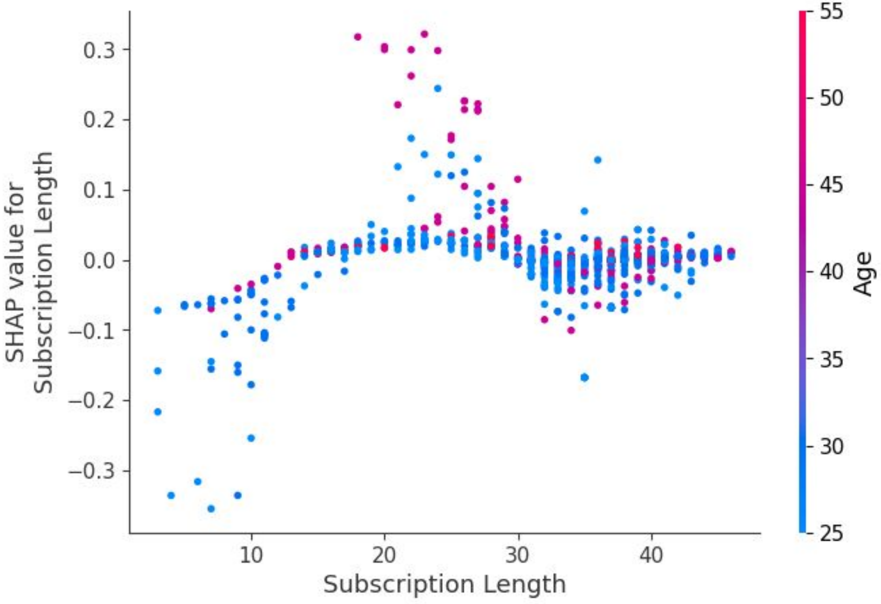
Summary Plot



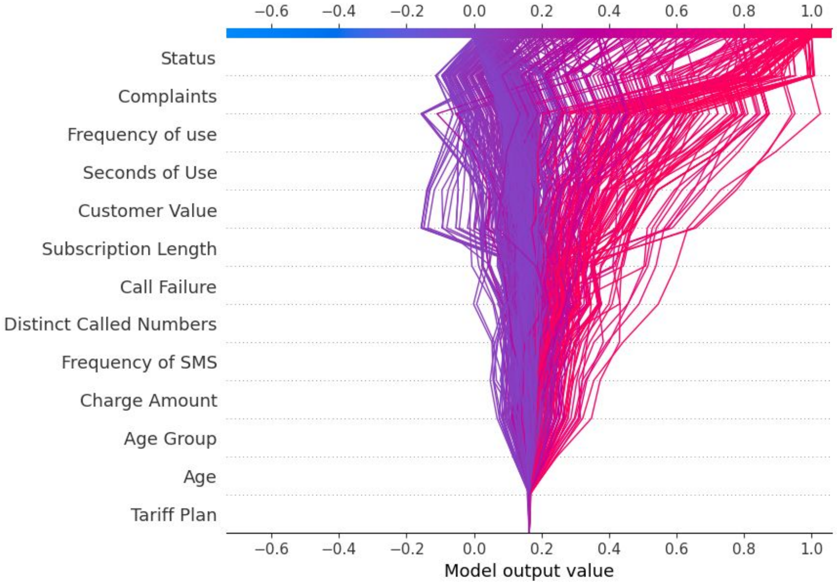
Other Summary Plot

Force Plot





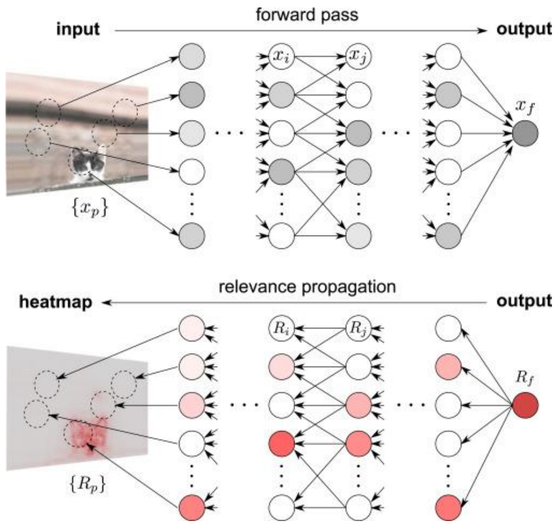
Dependence Plot For Subscription Length



Decision Plot

Layer-wise Relevance Propagation (LRP)

- LRP creates local explanations specifically for neural networks
- It follows this general set of steps to create explanations



Layer-wise Relevance Propagation (LRP)

- LRP creates local explanations specifically for neural networks
- It follows this general set of steps to create explanations
  - Forward pass a sample through the neural network (create a prediction)

Layer-wise Relevance Propagation (LRP)

- LRP creates local explanations specifically for neural networks
- It follows this general set of steps to create explanations
  - Forward pass a sample through the neural network (create a prediction)
  - Assign a relevance score equal to the prediction value

Layer-wise Relevance Propagation (LRP)

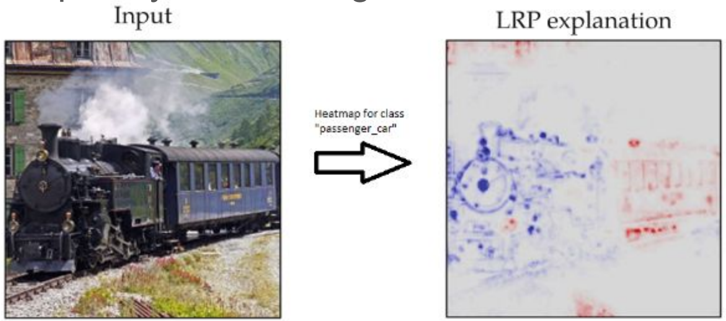
- LRP creates local explanations specifically for neural networks
- It follows this general set of steps to create explanations
  - Forward pass a sample through the neural network (create a prediction)
  - Assign a relevance score equal to the prediction value
  - Relevance score is propagated back through the network
    - Each layer redistributes the relevance it received among its neurons proportional to their contribution

Layer-wise Relevance Propagation (LRP)

- LRP creates local explanations specifically for neural networks
- It follows this general set of steps to create explanations
    - Forward pass a sample through the neural network (create a prediction)
    - Assign a relevance score equal to the prediction value
    - Relevance score is propagated back through the network
        - Each layer redistributes the relevance it received among its neurons proportional to their contribution
        - There are redistribution rules used to determine how this happens

Layer-wise Relevance Propagation (LRP)

- LRP creates local explanations specifically for neural networks
- It follows this general set of steps to create explanations
  - Forward pass a sample through the neural network (create a prediction)
  - Assign a relevance score equal to the prediction value
  - Relevance score is propagated back through the network
    - Each layer redistributes the relevance it received among its neurons proportional to their contribution
    - There are redistribution rules used to determine how this happens
  - Propagate back to input layer and assign relevance scores



Quick Review

- Perturbation based explanations (LIME)
- Game-theory/Feature based explanations (SHAP)
- Decomposition based (LRP)

Quick Review

- Perturbation based explanations (LIME)
- Game-theory/Feature based explanations (SHAP)
- Decomposition based (LRP)

- Python packages
  - LIME (pip install lime)
  - SHAP (pip install shap)
  - LRP (no simple pip install, there are other implementations it seems)
  - ELI5 (pip install eli5)
  - Interpret (pip install interpret) – appears to be an explanation suite
  - OmniXAI (pip install omnixai) – appears to be an explanation suite

The Counterfactual

- Local explanation centered around flipping the label
  - “Explanation describes the smallest change to the feature values that changes the prediction to a predefined output”
- Becoming more common in the literature (especially with the rise of LLMs)

The Counterfactual

- Local explanation centered around flipping the label
  - “Explanation describes the smallest change to the feature values that changes the prediction to a predefined output”
- Becoming more common in the literature (especially with the rise of LLMs)
- General Algorithm:
  - Select a sample to be explained and the desired label to change to (y’)

The Counterfactual

- Local explanation centered around flipping the label
  - “Explanation describes the smallest change to the feature values that changes the prediction to a predefined output”
- Becoming more common in the literature (especially with the rise of LLMs)
- General Algorithm:
  - Select a sample to be explained and the desired label to change to (y’)
  - Sample a random instance as the initial counterfactual

The Counterfactual

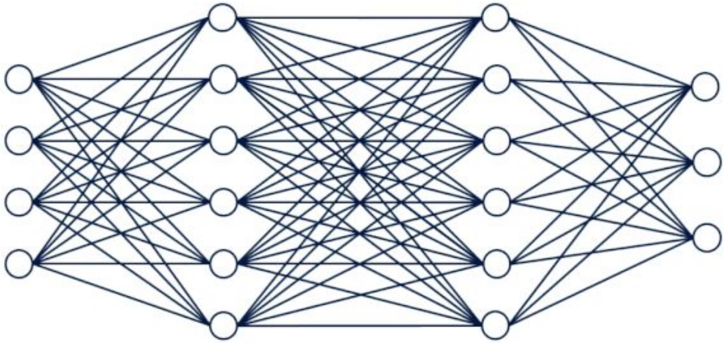
- Local explanation centered around flipping the label
    - "Explanation describes the smallest change to the feature values that changes the prediction to a predefined output"
- Becoming more common in the literature (especially with the rise of LLMs)
- General Algorithm:
    - Select a sample to be explained and the desired label to change to (y')
    - Sample a random instance as the initial counterfactual
    - Optimize a loss function with the above counterfactual as the starting point
        - -> $L(x, x', y', \lambda) = \lambda \cdot (\hat{f}(x') - y')^2 + d(x, x')$

The Counterfactual

- Local explanation centered around flipping the label
  - "Explanation describes the smallest change to the feature values that changes the prediction to a predefined output"
- Becoming more common in the literature (especially with the rise of LLMs)
- General Algorithm:
  - Select a sample to be explained and the desired label to change to (y')
  - Sample a random instance as the initial counterfactual
  - Optimize a loss function with the above counterfactual as the starting point
    - -> $L(x, x', y', \lambda) = \lambda \cdot (\hat{f}(x') - y')^2 + d(x, x')$
    - $(\hat{f}(x') - y')^2$ – quadratic distance between counterfactual and desired output
    - $d(x, x')$ – distance between original and counterfactual samples
    - λ – affects explanations. Higher values prefers predictions close to y'. Lower values prefer counterfactuals x' more similar to x

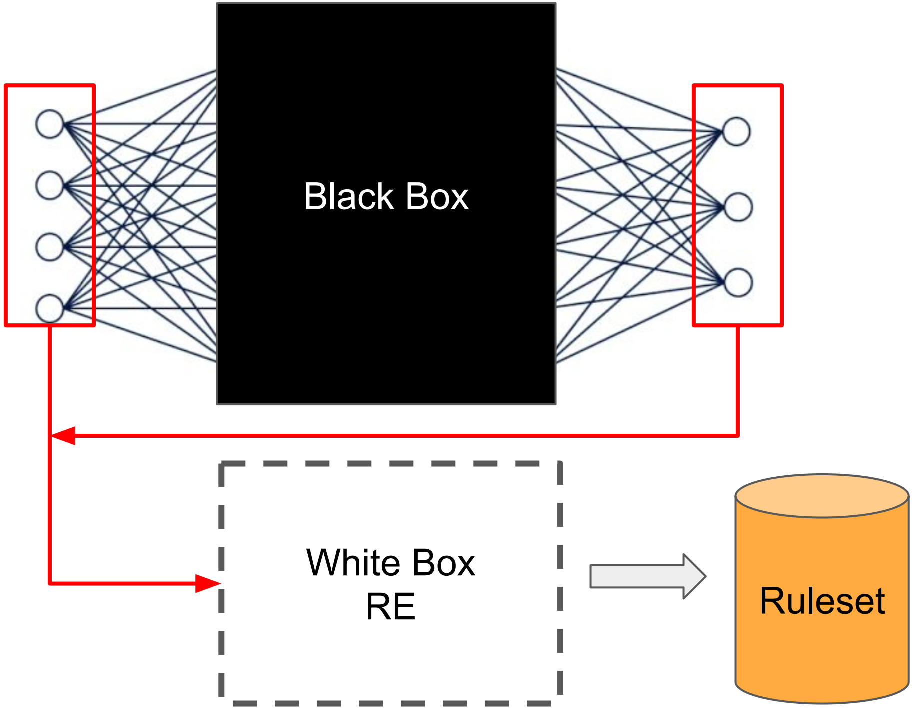
Rule Extraction

- Types
  - Pedagogical
  - Decompositional
  - Eclectic



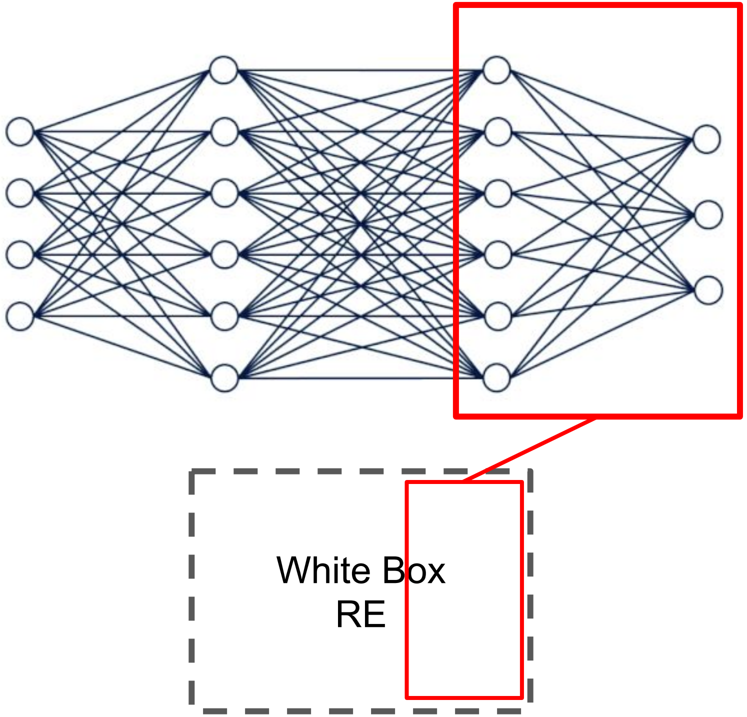
Rule Extraction

- **Pedagogical**
  - Inputs and outputs
  - Train a decision tree
  - **Maintains the black box**
  - Trustworthy?



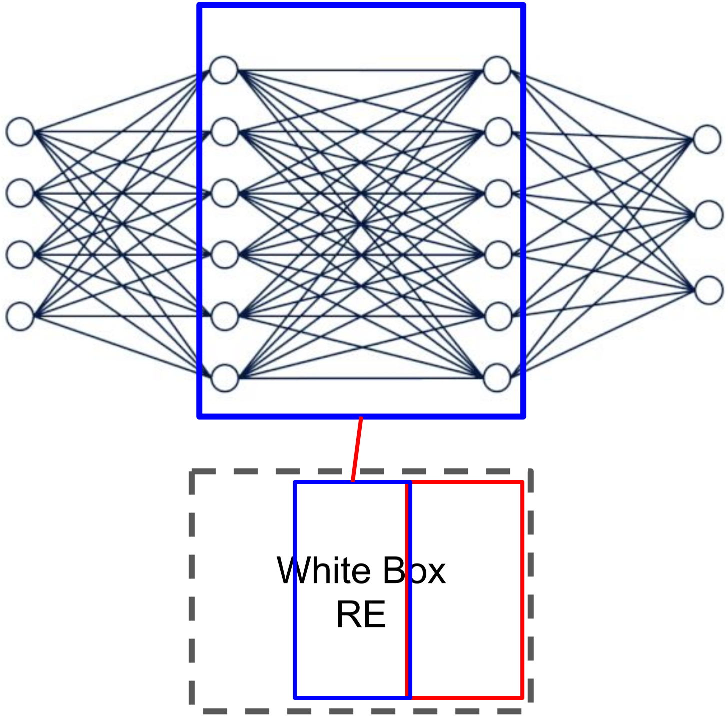
Rule Extraction

- **Decompositional**
  - Uses weights
  - Trains a decision tree per layer
  - **Opens the black box**
  - Expensive, but trustworthy



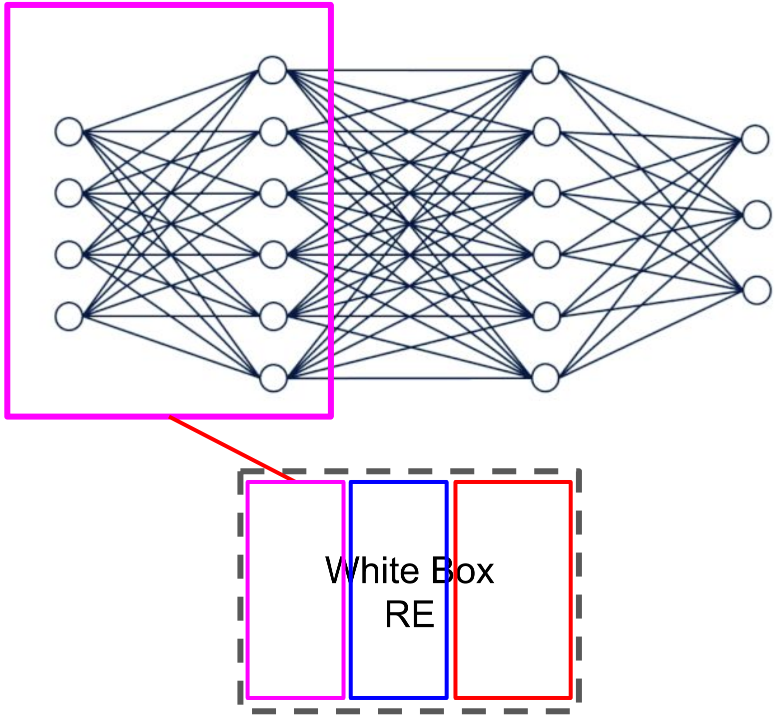
Rule Extraction

- **Decompositional**
  - Uses weights
  - Trains a decision tree per layer
  - **Opens the black box**
  - Expensive, but trustworthy



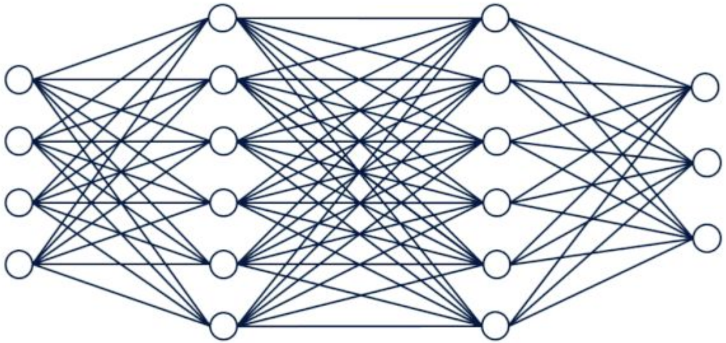
Rule Extraction

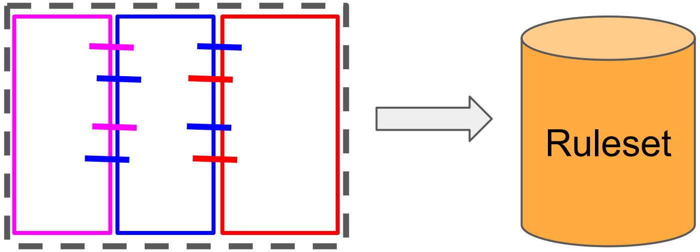
- **Decompositional**
    - Uses weights
    - Trains a decision tree per layer
    - **Opens the black box**
    - Expensive, but trustworthy



Rule Extraction

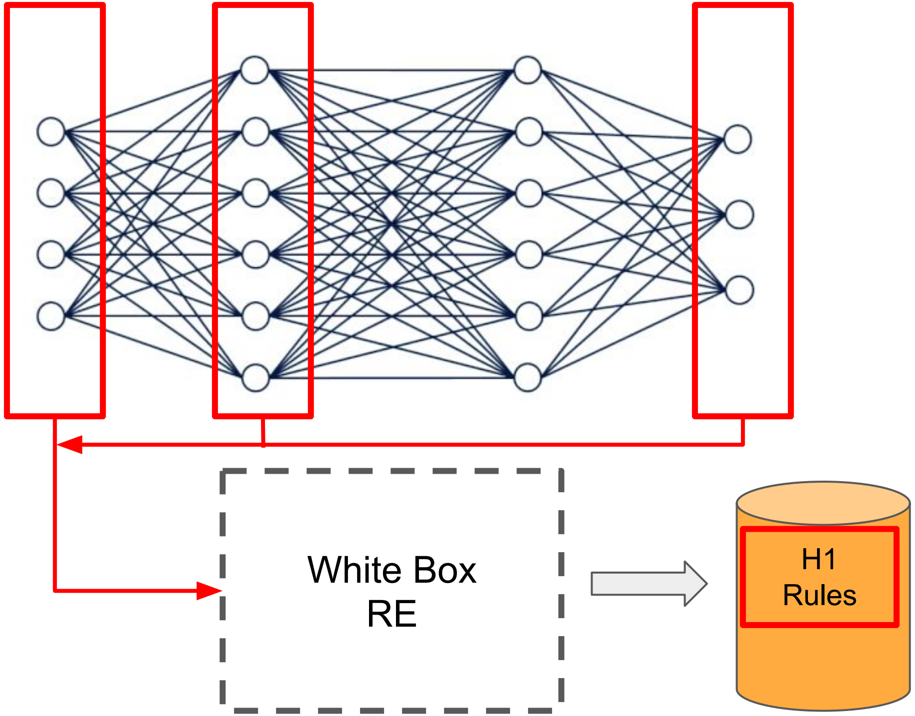
- **Decompositional**
  - Uses weights
  - Trains a decision tree per layer
  - **Opens the black box**
  - Expensive, but trustworthy





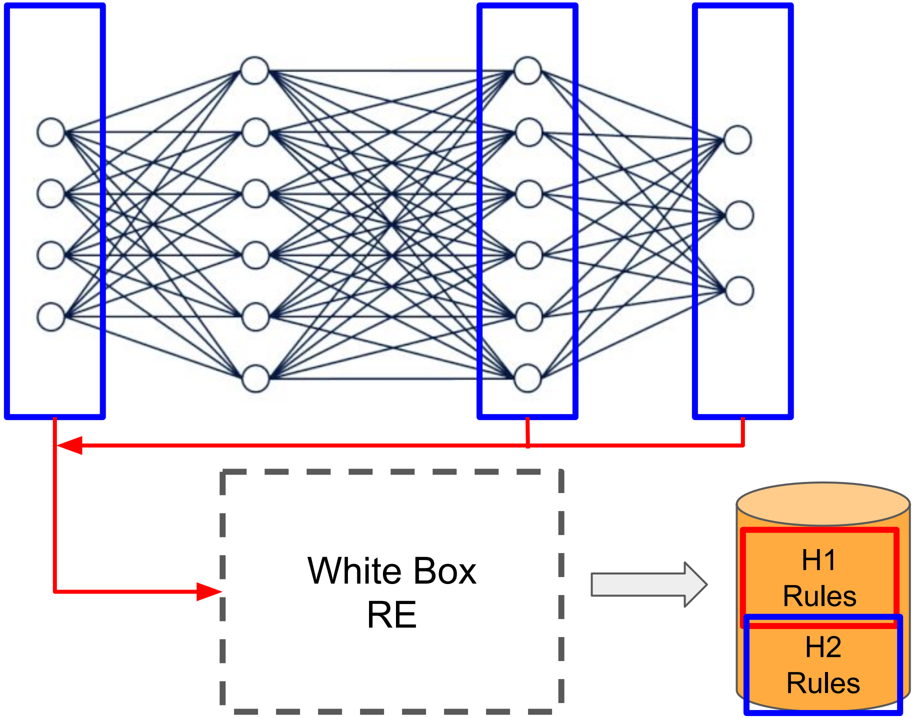
Rule Extraction

- **Eclectic**
  - Uses Inputs, weights, outputs
  - Extracts rules per layer
  - **Opens the black box**
  - Not as Expensive, but trustworthy



Rule Extraction

- **Eclectic**
  - Uses Inputs, weights, outputs
  - Extracts rules per layer
  - **Opens the black box**
  - Not as Expensive, but trustworthy



Rule Extraction

Term

Feature Threshold

Petal Width (cm) > .6 && Petal Length (cm) <= 4.75 && Petal Width (cm) <= 1.6 && Sepal Width (cm) <= 3.15 == versicolor

Label

Rule

Ruleset

```
petal width (cm) <= 0.80 == setosa
petal width (cm) > 0.60 && petal length (cm) <= 4.75 && petal width (cm) <= 1.60 && sepal width (cm) <= 3.15 == versicolor
petal length (cm) <= 2.45 == setosa
petal width (cm) > 0.60 && petal length (cm) <= 4.75 && petal width (cm) <= 1.60 && sepal width (cm) <= 3.15 == versicolor
petal length (cm) > 4.75 && sepal width (cm) <= 2.90 && petal width (cm) <= 1.75 == virginica
petal length (cm) > 4.75 && petal width (cm) > 1.75 == virginica
petal length (cm) > 4.75 && sepal width (cm) <= 2.60 && petal length (cm) <= 5.05 && petal width (cm) <= 1.75 == virginica
petal length (cm) > 4.75 && sepal width (cm) > 2.90 && sepal length (cm) > 7.05 && petal width (cm) <= 1.75 == virginica
petal width (cm) > 1.60 && petal length (cm) <= 4.75 == virginica
petal length (cm) > 4.75 && petal width (cm) > 1.75 == virginica
petal width (cm) > 1.60 && petal length (cm) <= 4.75 == virginica
petal length (cm) > 4.75 && petal length (cm) > 5.05 && petal width (cm) <= 1.75 == virginica
sepal width (cm) <= 3.15 && sepal width (cm) > 2.90 && petal length (cm) > 4.75 && petal length (cm) <= 5.40 && petal width (cm) <= 1.75 == versicolor
sepal width (cm) <= 3.15 && sepal length (cm) > 6.50 && petal length (cm) > 4.75 && petal length (cm) <= 5.40 && petal width (cm) <= 1.75 == versicolor
sepal length (cm) > 6.35 && sepal width (cm) > 3.15 && petal width (cm) <= 1.65 == versicolor
petal width (cm) > 0.80 && sepal width (cm) > 3.15 && petal length (cm) <= 4.75 && sepal width (cm) <= 3.25 == versicolor
```

Eclectic Rule Extraction Algorithm

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$Y' =$ M.predict(X)

for hidden layer h_i in M **do**

  $X' = h_i(X)$

  $hidden_{dt} = DT_{alg}(X', Y')$

  $Rules_{hidden} =$ ExtractRules($hidden_{dt}$)

  $\hat{Y} = Rules_{hidden}(X')$

  $input_{dt} = DT_{alg}(X, \hat{Y})$

  $R.add$(ExtractRules($input_{dt}$))

end for

return R

END

Eclectic Rule Extraction Algorithm

- Input a dataset, model, and decision tree algorithm

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$Y' =$ M.predict(X)

for hidden layer h_i in M **do**

  $X' = h_i(X)$

  $hidden_{dt} = DT_{alg}(X', Y')$

  $Rules_{hidden} = \text{ExtractRules}(hidden_{dt})$

  $\hat{Y} = Rules_{hidden}(X')$

  $input_{dt} = DT_{alg}(X, \hat{Y})$

  $R.add(\text{ExtractRules}(input_{dt}))$

end for

return R

END

Eclectic Rule Extraction Algorithm

- Input a dataset, model, and decision tree algorithm
- Initialize a set to save rulesets

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$Y' =$ M.predict(X)

for hidden layer h_i in M **do**

  $X' = h_i(X)$

  $hidden_{dt} = DT_{alg}(X', Y')$

  $Rules_{hidden} = \text{ExtractRules}(hidden_{dt})$

  $\hat{Y} = Rules_{hidden}(X')$

  $input_{dt} = DT_{alg}(X, \hat{Y})$

  $R.add(\text{ExtractRules}(input_{dt}))$

end for

return R

END

Eclectic Rule Extraction Algorithm

- Input a dataset, model, and decision tree algorithm
- Initialize a set to save rulesets
- Get dataset predictions from model

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$\boxed{Y' = \text{M.predict(X)}}$

for hidden layer h_i in M **do**

  $X' = h_i(X)$

  $hidden_{dt} = DT_{alg}(X', Y')$

  $Rules_{hidden} = \text{ExtractRules}(hidden_{dt})$

  $\hat{Y} = Rules_{hidden}(X')$

  $input_{dt} = DT_{alg}(X, \hat{Y})$

  $R.add(\text{ExtractRules}(input_{dt}))$

end for

return R

END

Eclectic Rule Extraction Algorithm

- For each hidden layer:

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$Y' =$ M.predict(X)

for hidden layer h_i in M **do**

  $X' = h_i(X)$

  $hidden_{dt} = DT_{alg}(X', Y')$

  $Rules_{hidden} = \text{ExtractRules}(hidden_{dt})$

  $\hat{Y} = Rules_{hidden}(X')$

  $input_{dt} = DT_{alg}(X, \hat{Y})$

  $R.add(\text{ExtractRules}(input_{dt}))$

end for

return R

END

Eclectic Rule Extraction Algorithm

- For each hidden layer:
  - Transform samples in X using hidden neuron values

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$Y' =$ M.predict(X)

for hidden layer h_i in M **do**

  $\boxed{X' = h_i(X)}$

  $hidden_{dt} = DT_{alg}(X', Y')$

  $Rules_{hidden} = \text{ExtractRules}(hidden_{dt})$

  $\hat{Y} = Rules_{hidden}(X')$

  $input_{dt} = DT_{alg}(X, \hat{Y})$

  $R.add(\text{ExtractRules}(input_{dt}))$

end for

return R

END

Eclectic Rule Extraction Algorithm

- For each hidden layer:
  - Transform samples in X using hidden neuron values
  - Train a hidden layer decision tree

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$Y' =$ M.predict(X)

for hidden layer h_i in M **do**

    $X' = h_i(X)$

    $hidden_{dt} = DT_{alg}(X', Y')$

    $Rules_{hidden} = \text{ExtractRules}(hidden_{dt})$

    $\hat{Y} = Rules_{hidden}(X')$

    $input_{dt} = DT_{alg}(X, \hat{Y})$

    $R.add(\text{ExtractRules}(input_{dt}))$

end for

return R

END

Eclectic Rule Extraction Algorithm

- For each hidden layer:
  - Transform samples in X using hidden neuron values
  - Train a hidden layer decision tree
  - Extract rules from the trained hidden layer decision tree

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$Y' =$ M.predict(X)

for hidden layer h_i in M **do**

    $X' = h_i(X)$

    $hidden_{dt} = DT_{alg}(X', Y')$

    $\boxed{Rules_{hidden} = \text{ExtractRules}(hidden_{dt})}$

    $\hat{Y} = Rules_{hidden}(X')$

    $input_{dt} = DT_{alg}(X, \hat{Y})$

    $R.add(\text{ExtractRules}(input_{dt}))$

end for

return R

END

Eclectic Rule Extraction Algorithm

- For each hidden layer:
  - Transform samples in X using hidden neuron values
  - Train a hidden layer decision tree
  - Extract rules from the trained hidden layer decision tree
  - Predict using the hidden layer ruleset

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$Y' =$ M.predict(X)

for hidden layer h_i **in** M **do**

$\quad X' = h_i(X)$

$\quad hidden_{dt} = DT_{alg}(X', Y')$

$\quad Rules_{hidden} = \text{ExtractRules}(hidden_{dt})$

$\quad \boxed{\hat{Y} = Rules_{hidden}(X')}$

$\quad input_{dt} = DT_{alg}(X, \hat{Y})$

$\quad R.add(\text{ExtractRules}(input_{dt}))$

end for

return R

END

Eclectic Rule Extraction Algorithm

- For each hidden layer:
  - Transform samples in X using hidden neuron values
  - Train a hidden layer decision tree
  - Extract rules from the trained hidden layer decision tree
  - Predict using the hidden layer ruleset
  - Train input DT using hidden layer predictions and original dataset

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$Y' =$ M.predict(X)

for hidden layer h_i **in** M **do**

  $X' = h_i(X)$

  $hidden_{dt} = DT_{alg}(X', Y')$

  $Rules_{hidden} = \text{ExtractRules}(hidden_{dt})$

  $\hat{Y} = Rules_{hidden}(X')$

  $input_{dt} = DT_{alg}(X, \hat{Y})$

  $R.add(\text{ExtractRules}(input_{dt}))$

end for

return R

END

Eclectic Rule Extraction Algorithm

- For each hidden layer:
  - Transform samples in X using hidden neuron values
  - Train a hidden layer decision tree
  - Extract rules from the trained hidden layer decision tree
  - Predict using the hidden layer ruleset
  - Train input DT using hidden layer predictions and original dataset
  - Extract rules from the input DT and add them to the ruleset

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$Y' =$ M.predict(X)

for hidden layer h_i in M **do**

  $X' = h_i(X)$

  $hidden_{dt} = DT_{alg}(X', Y')$

  $Rules_{hidden} = \text{ExtractRules}(hidden_{dt})$

  $\hat{Y} = Rules_{hidden}(X')$

  $input_{dt} = DT_{alg}(X, \hat{Y})$

  $\boxed{R.add(\text{ExtractRules}(input_{dt}))}$

end for

return R

END

Eclectic Rule Extraction Algorithm

- For each hidden layer:
  - Transform samples in X using hidden neuron values
  - Train a hidden layer decision tree
  - Extract rules from the trained hidden layer decision tree
  - Predict using the hidden layer ruleset
  - Train input DT using hidden layer predictions and original dataset
  - Extract rules from the input DT and add them to the ruleset
- Return the final ruleset

Algorithm 1 DNN Rule Extraction

Input: Dataset (X), Model (M), Decision Tree Algorithm (DT_{alg})

Output: Final Ruleset (R)

BEGIN

$R = set()$

$Y' =$ M.predict(X)

for hidden layer h_i in M **do**

 $X' = h_i(X)$

 $hidden_{dt} = DT_{alg}(X', Y')$

 $Rules_{hidden} = \mathrm{ExtractRules}(hidden_{dt})$

 $\hat{Y} = Rules_{hidden}(X')$

 $input_{dt} = DT_{alg}(X, \hat{Y})$

 $R.add(\mathrm{ExtractRules}(input_{dt}))$

end for

return R

END