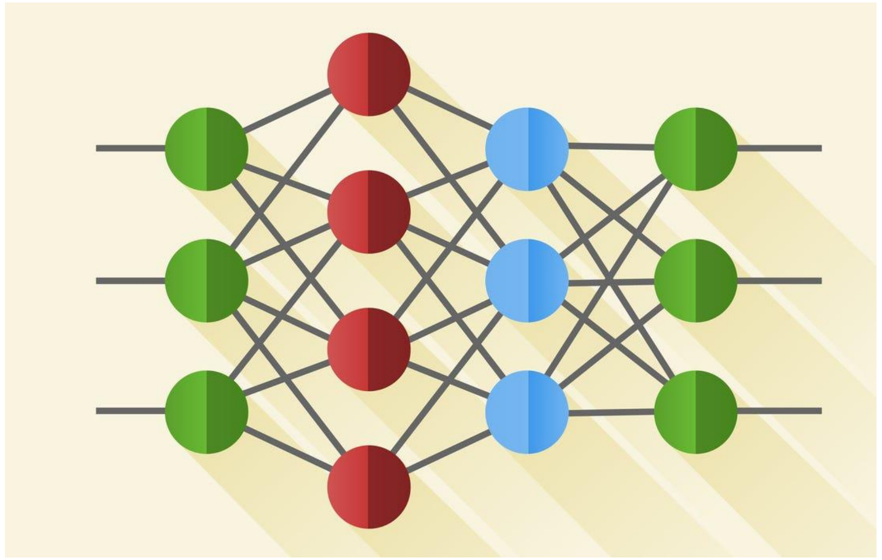
Session 6: Explainability & Neural Networks



Prof. Carlos Cano Domingo
c.cano-domingo@tbs-education.org
Session 6

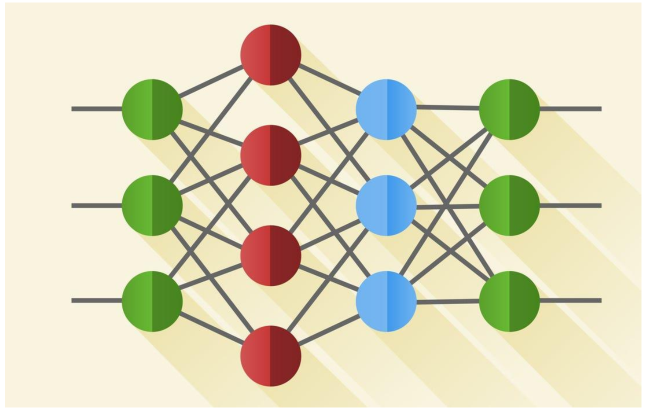
Content

Session 6: Explainability & Neural Networks

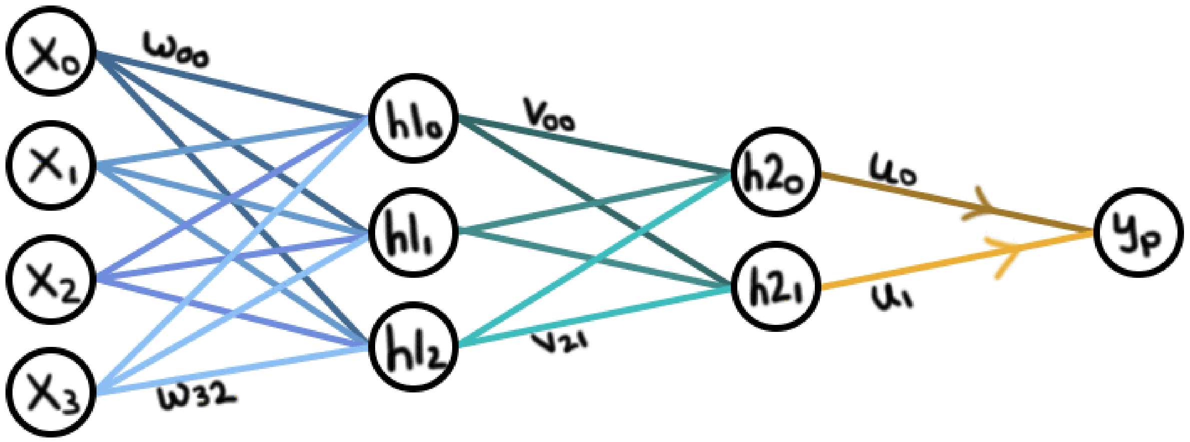
Objective: Introduce Explainable ML and a brief introduction to Neural networks

Content:
- Explainable ML
- Responsible ML
- Trustworthy ML
- Neural Network introduction

Business Focus: Explore the necessity of having Explainable ML to decision making ML approach, the importance of being conscious about the responsible ML and it trustworthiness. Applied a MLP to business data.



1. Explainable ML: Introduce concepts for understanding model decisions.
2. Responsible ML: Ensure models are fair, ethical, and accountable.
3. Trustworthy ML: Build reliable, unbiased models for business applications.
4. Neural Network Introduction: Brief overview of neural networks, layers, and activation functions focus on MLP.



Learning Outcome:

- Recognize the role of Explainable, Responsible, and Trustworthy ML.
- Understand the basics of Neural Networks in business contexts.

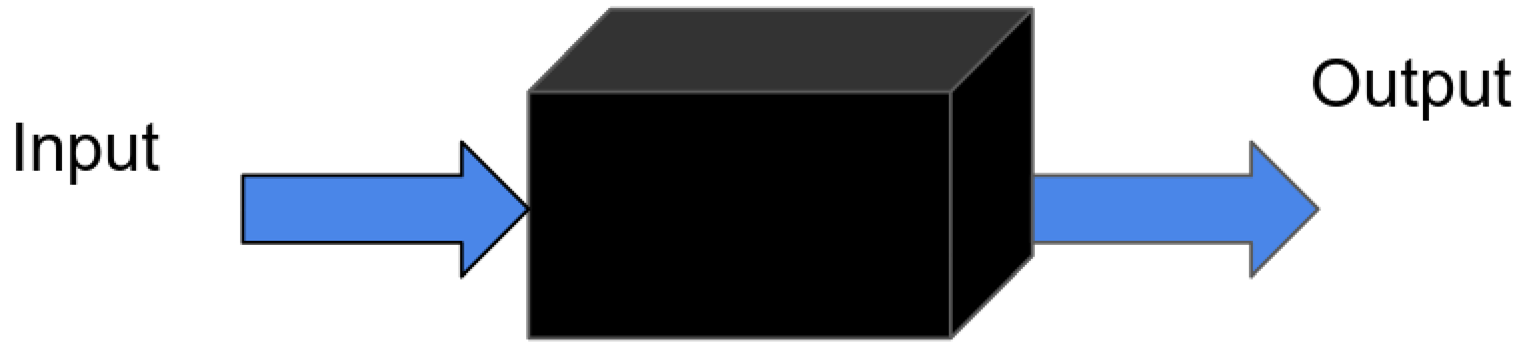
1 Explainable ML

Why do we need model explainability

- Use Machine Learning to review resumes:
  - Based on your capability or gender?
- Use Machine Learning to detect fraud transactions?
  - Why does the model think this transaction is suspicious?
- The above examples all belong to high-stakes decisions. The decisions have a huge impact on human well-being.

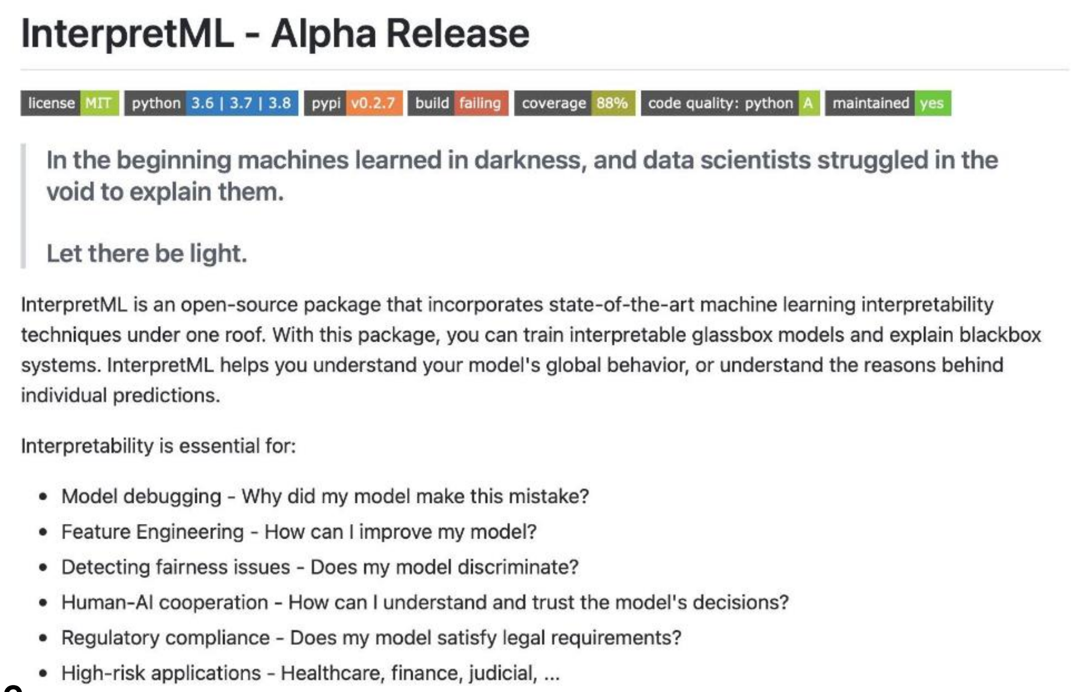
Black-box Model

- If the ML system is deployed in high-stakes decisions environment:
  - Is accuracy important?
  - Can we trust the machine learning model?
- In banking, insurance and other heavily regulated industries, model interpretability is a serious legal mandate
- In lots of critical areas such as healthcare, government, bioinformatics, etcs, rationale for models' decision is necessary for trust



Goals of Interpretability

- Model debugging
  - Why did my model make mistake?
- Feature Engineering
  - How can I improve my model?
- Detecting fairness issues
  - Does my model have biases?
- Human-AI cooperation
  - How can I understand and trust model's decision?
- Regulatory Compliance
  - Does my model satisfy legal requirements?
- High-stake Decisions
  - Healthcare, Finance..



InterpretML - Alpha Release

license MIT | python 3.6 | 3.7 | 3.8 | pypi v0.2.7 | build failing | coverage 88% | code quality: python A | maintained yes

> In the beginning machines learned in darkness, and data scientists struggled in the void to explain them.
>
> Let there be light.

InterpretML is an open-source package that incorporates state-of-the-art machine learning interpretability techniques under one roof. With this package, you can train interpretable glassbox models and explain blackbox systems. InterpretML helps you understand your model's global behavior, or understand the reasons behind individual predictions.

Interpretability is essential for:

- Model debugging - Why did my model make this mistake?
- Feature Engineering - How can I improve my model?
- Detecting fairness issues - Does my model discriminate?
- Human-AI cooperation - How can I understand and trust the model's decisions?
- Regulatory compliance - Does my model satisfy legal requirements?
- High-risk applications - Healthcare, finance, judicial, ...

Cae Study

Training Data

Digimon

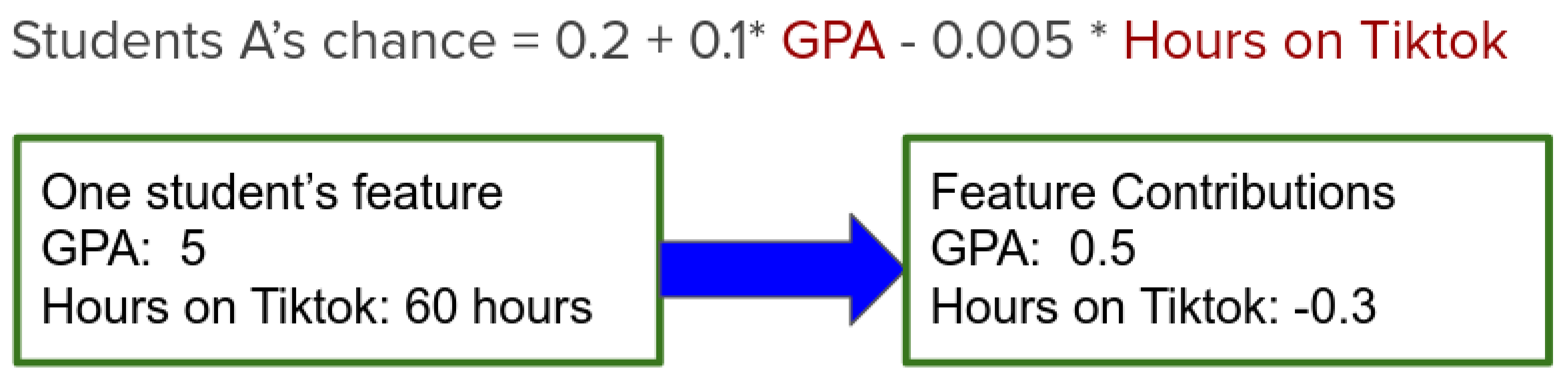
Pokemon

Testing Data

Digimon or Pokemon?

Linear models first

- Prediction is the linear combinations of the features values, weighted by the model coefficients



Students A's chance = 0.2 + 0.1* GPA - 0.005 * Hours on Tiktok

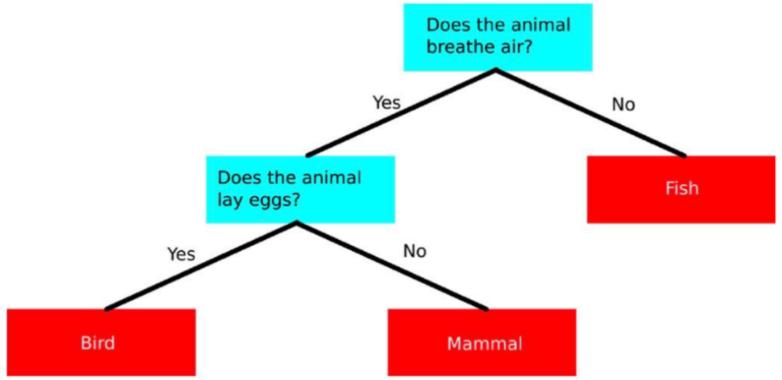
One student's feature
GPA: 5
Hours on Tiktok: 60 hours

Feature Contributions
GPA: 0.5
Hours on Tiktok: -0.3

- Capability of linear models is limited.

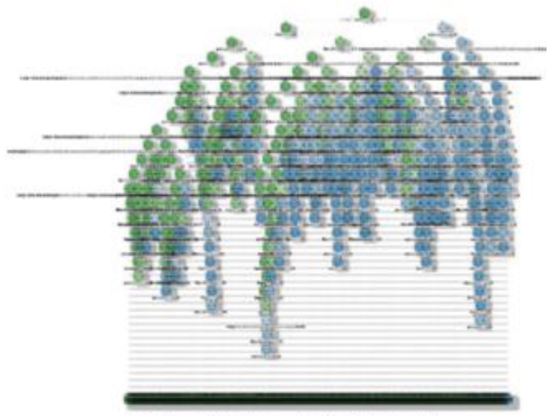
Decision tree

- It is “interpretable”.
- More powerful compared to linear models.



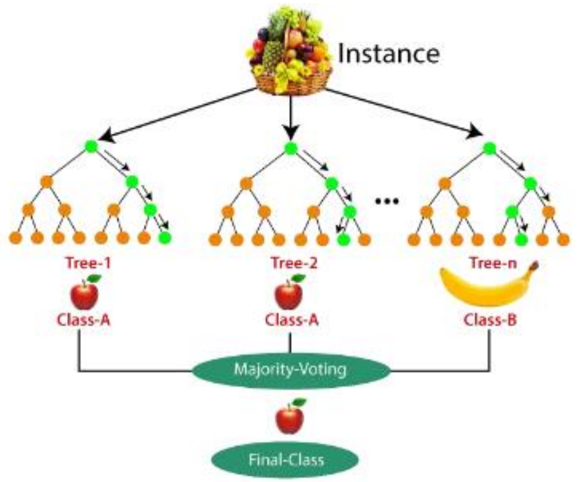
Decision tree can be complex

- It can be a huge and complex tree.

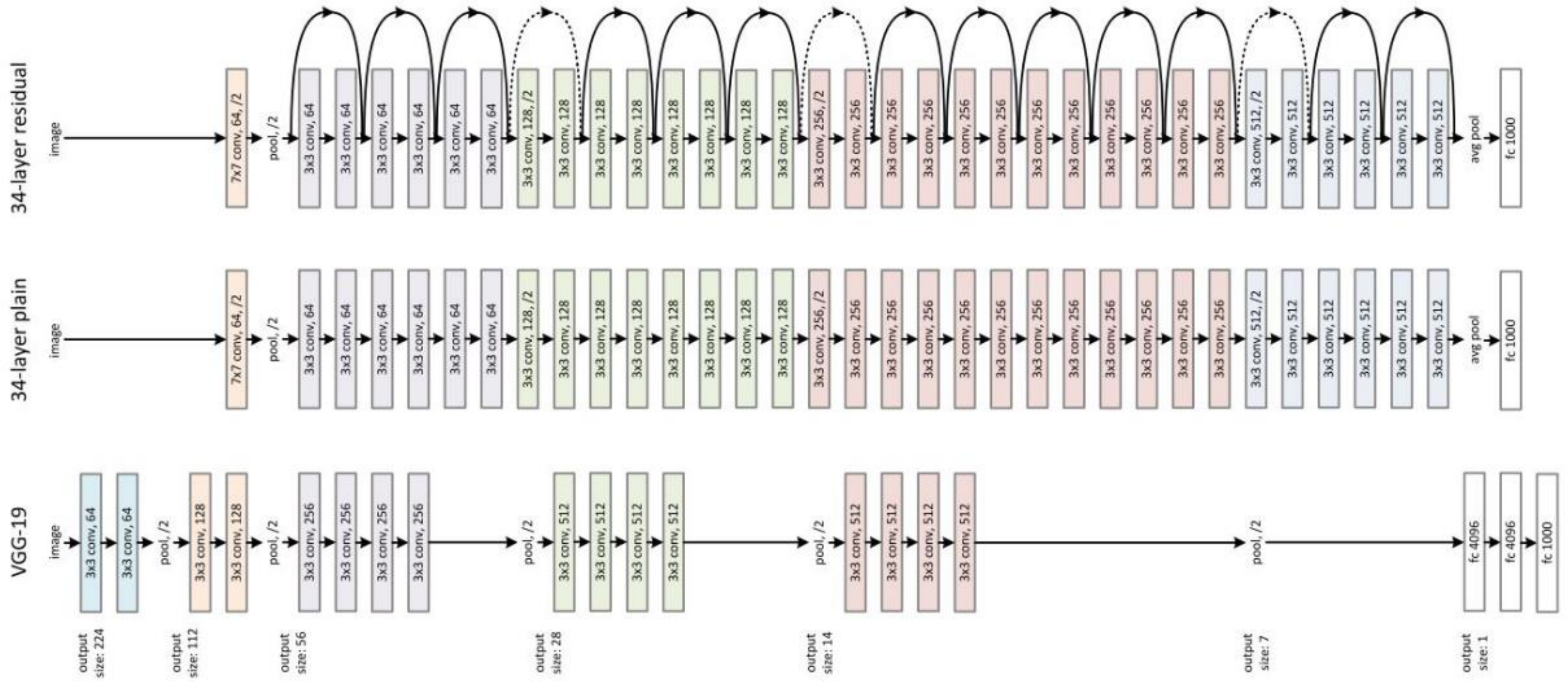


My goal is to extract some useful rules from the entire process

- It can be a forest.

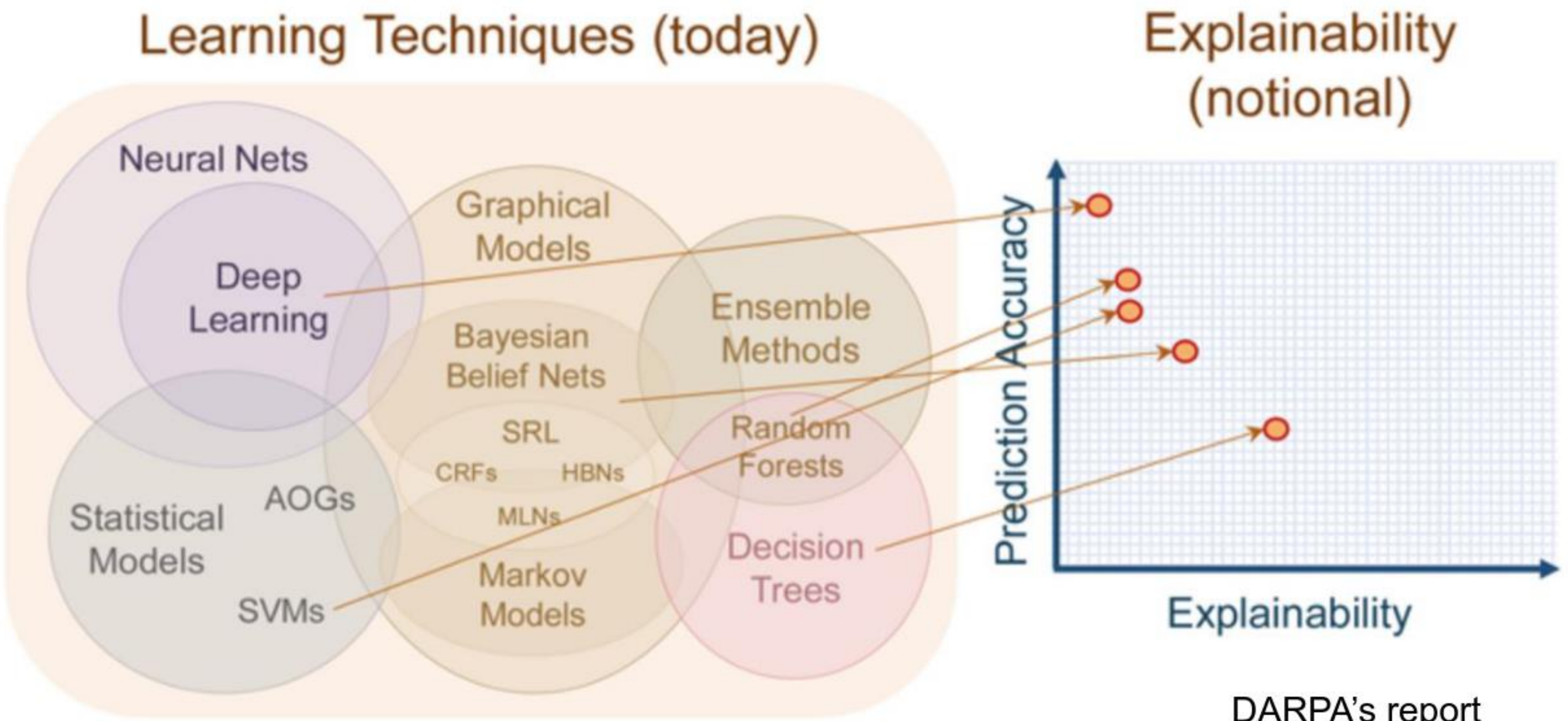


Complex models



For imagenet, they use 152 layers, which firstly achieved lower error rate compared to Humans in image recognition tasks.

Trade-off



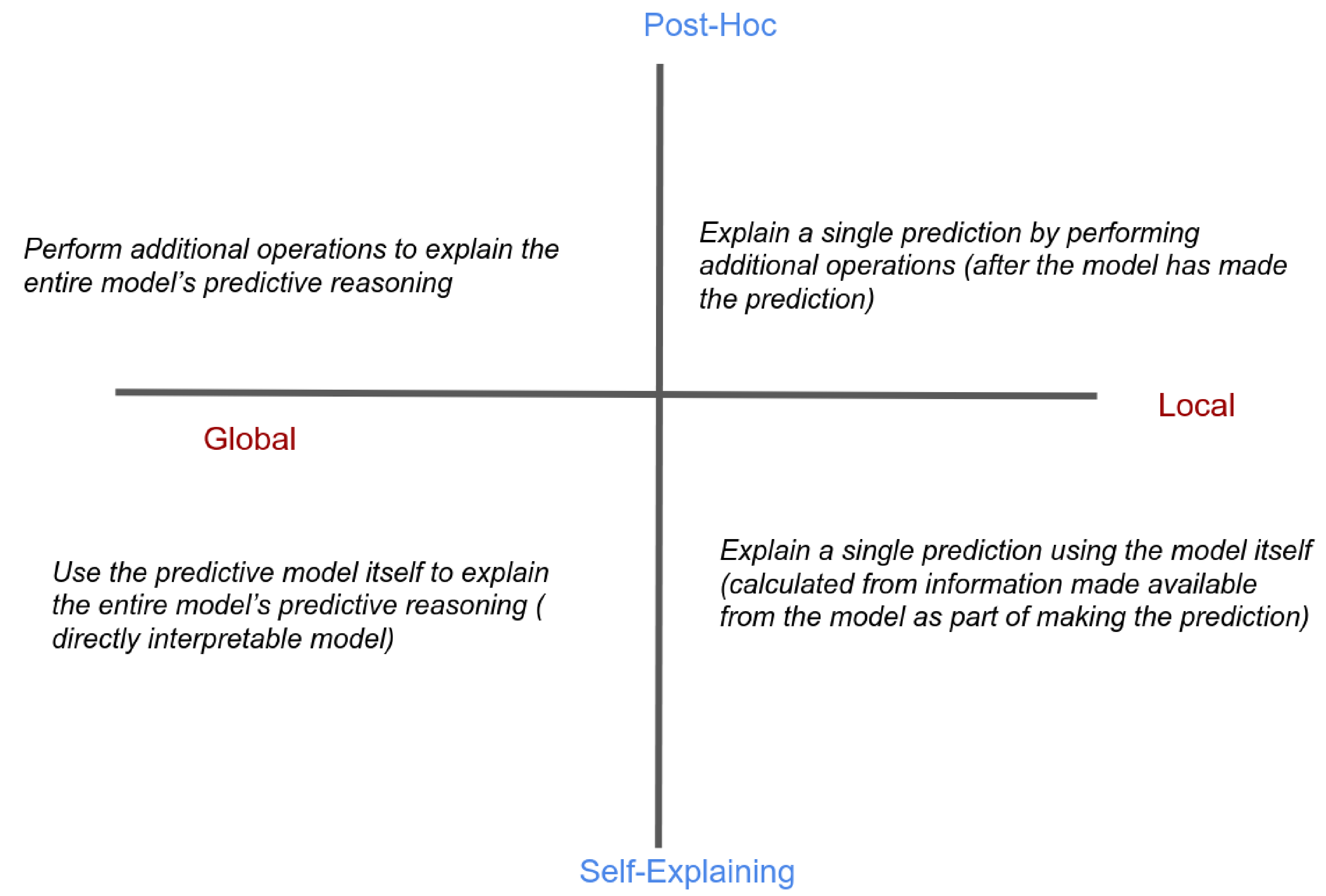
DARPA’s report

Decision tree

- Self-Explaining
  - Directly interpretable.
  - Generates the explanations at the same time as the prediction
  - Rule-based System, Decision Trees, Logistic Regression, Hidden Markov Model, etc.
- Post hoc:
  - Additional operation is performed after the predictions are made
  - Open-source packages: tf-keras-vis (gradient-based methods for deep learning), LIME, SHAP, etc

Categorization of interpretability

- Global
  - Explanation or justification by revealing how the model's predictive process works
  - What do you think pokemon looks like?
- Local:
  - Provide information or justification for the model's prediction on a specific input
  - Why do you think this image is pokemon?

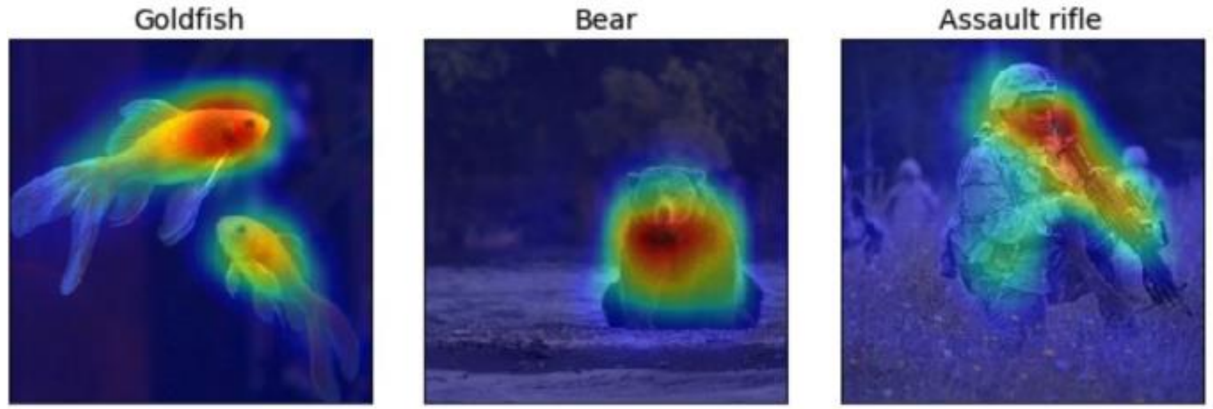

Post-Hoc
Perform additional operations to explain the entire model's predictive reasoning
Explain a single prediction by performing additional operations (after the model has made the prediction)
Local
Global
Use the predictive model itself to explain the entire model's predictive reasoning (directly interpretable model)
Explain a single prediction using the model itself (calculated from information made available from the model as part of making the prediction)
Self-Explaining

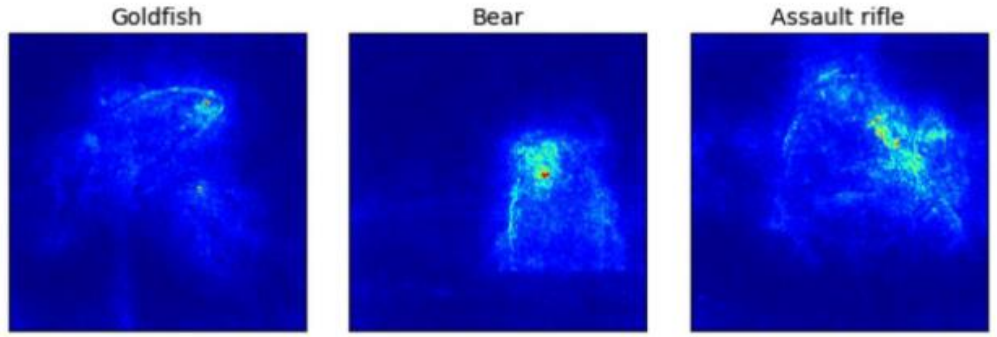
Example-driven

- Reasoning with examples
  - Explain the prediction of an input instance by identifying and presenting other instances
  - Eg. patient A has a tumor because he is similar to these k other data points with tumors
- Similar to nearest neighbor-based approaches

Feature importance

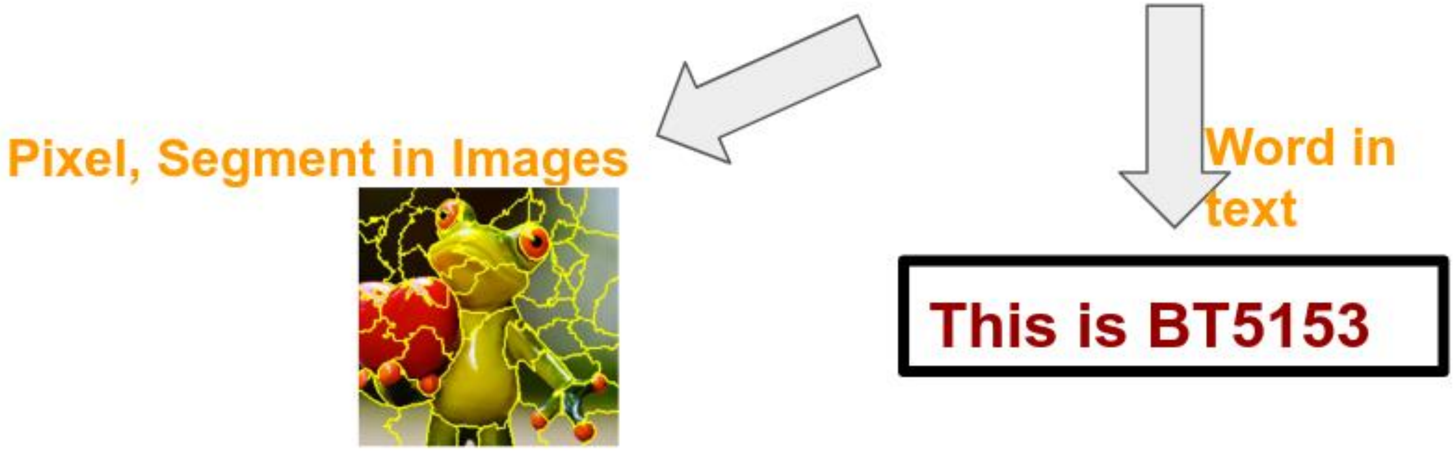
- Derive explanation by investigating the importance scores of different features used to output the final prediction
- It can be computed from
  - Attention Layer Approach
  - Gradient-based Saliency Approach





Gradient-based method

- Explain the decision made by the model
  - Eg, Why do you think this image is pokemon not digimon?
- Motivation: we want to know the contribution of each component/feature in the input data for prediction



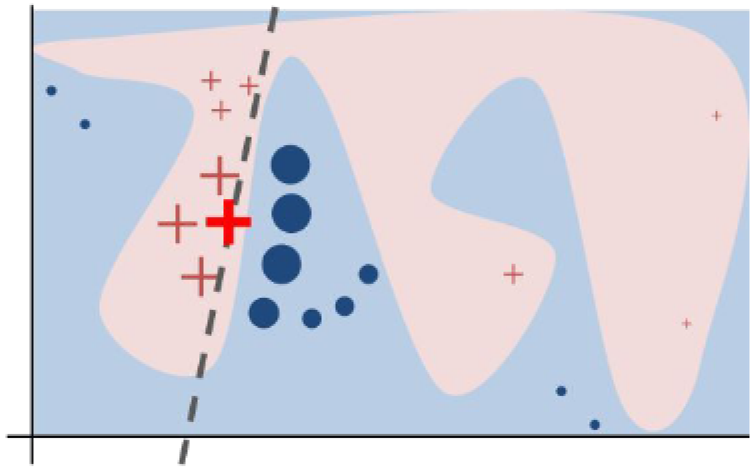
- Solution: Removing or modifying the partial parts of the components, observing the change of decision.

Surrogate model

- Model predictions are explained by learning a second, usually more explainable model, as a proxy
- Model-agnostic (applicable for any machine learning models)l prediction
- The learned surrogate models and the original models may have completely different mechanisms to make predictions

Surrogate model: local explanations

- Hard to explain a complex model in its entirety
  - How about explaining smaller regions?
  - Explain decisions of any model in a local region around a particular point
  - Learns sparse linear model

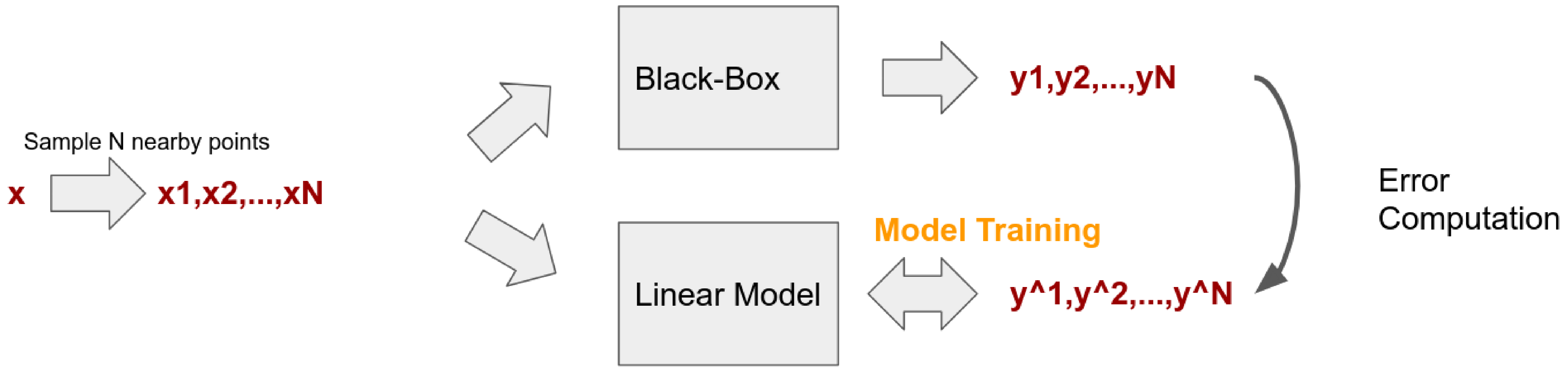


LIME (Ribeiro et. al)

Linear model can not mimic neural networks..but it may mimic a local region

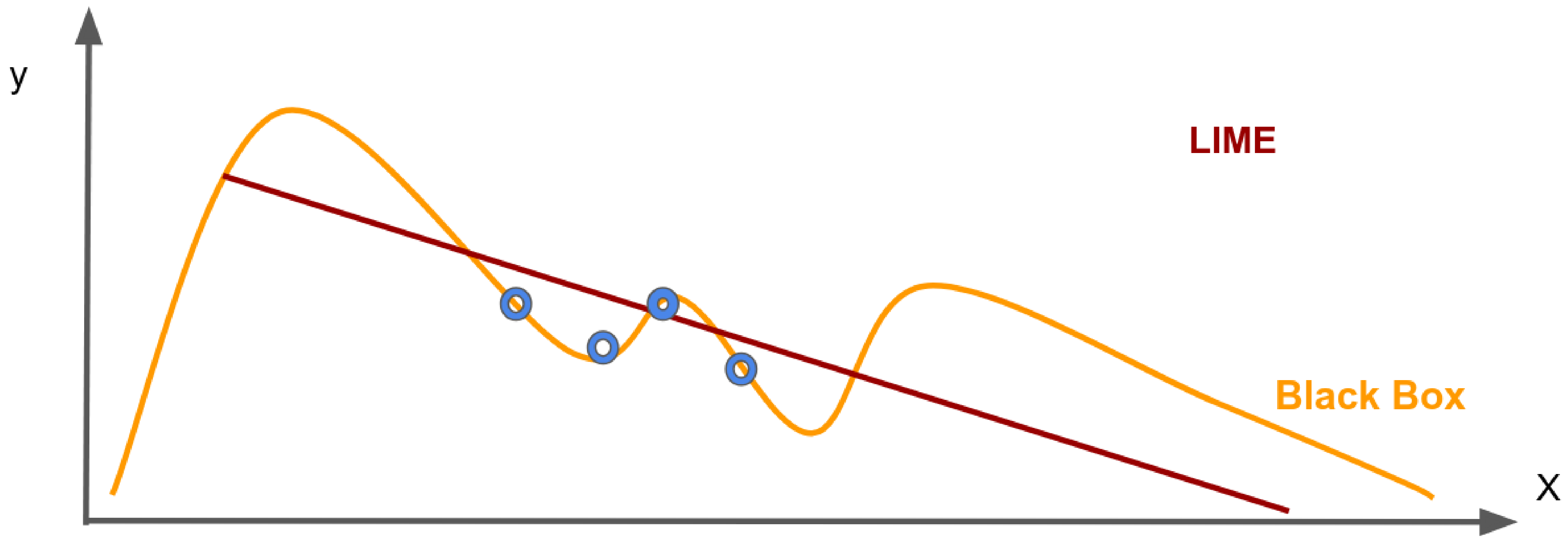
Surrogate model: local explanations

- Interpretable model can be used to mimic the actions of an complex model



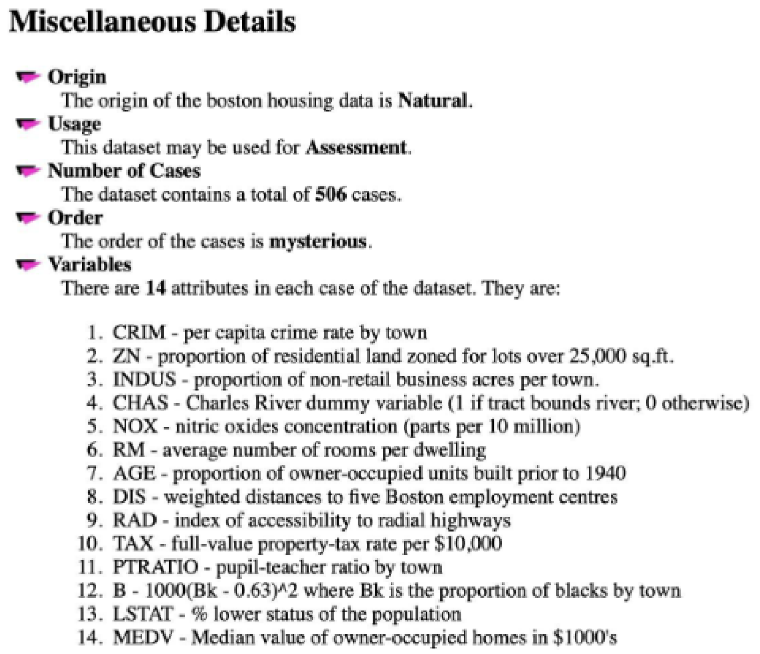
Local interpretable model-agnostic explanations

- Given a data point you want to explain
- Sample at the nearby
- Fit with linear model (or other interpretable models)
- Interpret the linear model



XGBoost’s feature importance

- XGBoost use boosting to combine weak learners to make accurate predictions.
- Feature importance for XGBoost could be checked in the following methods:
  - Built-in Function
  - Permutation method
  - SHAP method



Miscellaneous Details

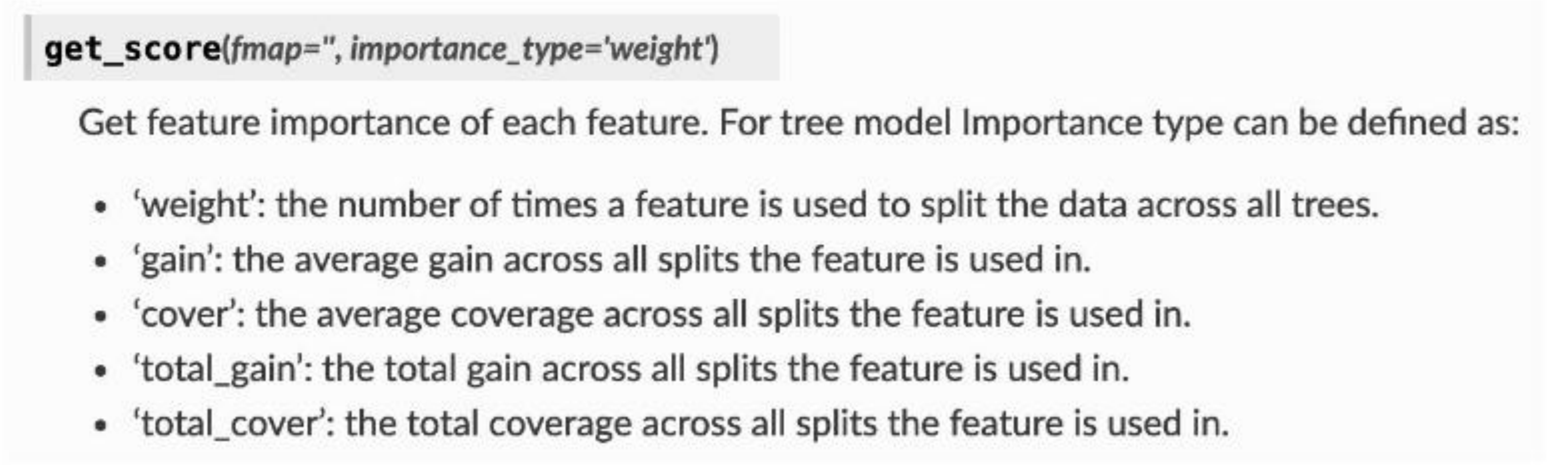
- **Origin**
  The origin of the boston housing data is **Natural**.
- **Usage**
  This dataset may be used for **Assessment**.
- **Number of Cases**
  The dataset contains a total of **506** cases.
- **Order**
  The order of the cases is **mysterious**.
- **Variables**
  There are **14** attributes in each case of the dataset. They are:

  1. CRIM - per capita crime rate by town
  2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
  3. INDUS - proportion of non-retail business acres per town.
  4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
  5. NOX - nitric oxides concentration (parts per 10 million)
  6. RM - average number of rooms per dwelling
  7. AGE - proportion of owner-occupied units built prior to 1940
  8. DIS - weighted distances to five Boston employment centres
  9. RAD - index of accessibility to radial highways
  10. TAX - full-value property-tax rate per $10,000
  11. PTRATIO - pupil-teacher ratio by town
  12. B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
  13. LSTAT - % lower status of the population
  14. MEDV - Median value of owner-occupied homes in $1000's

Housing price prediction

XGBoost built-in function

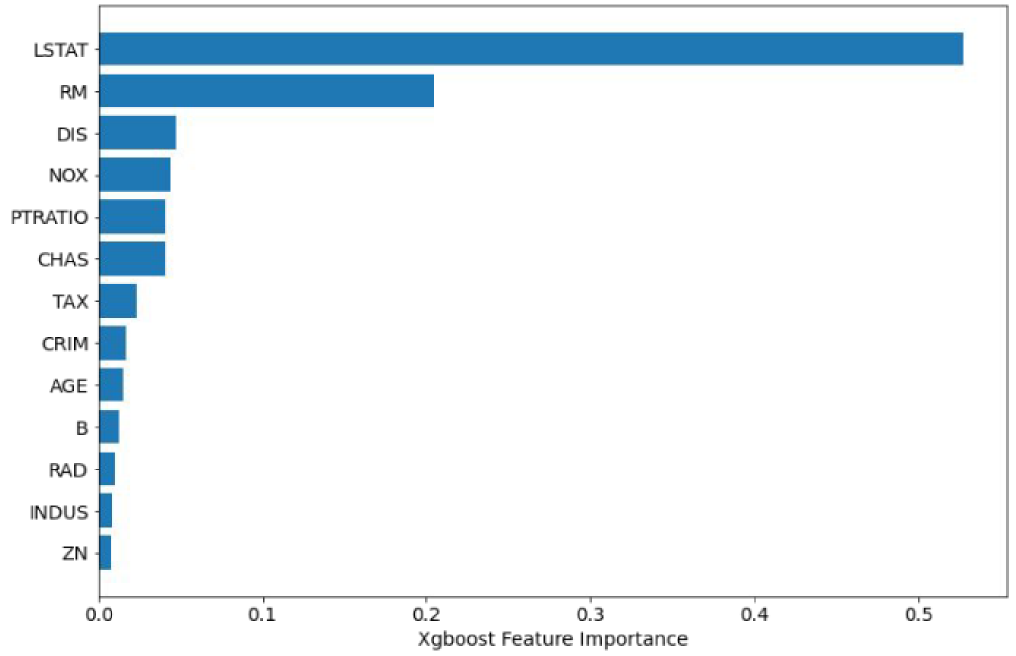
- XGBoost



get_score(*fmap='', importance_type='weight'*)

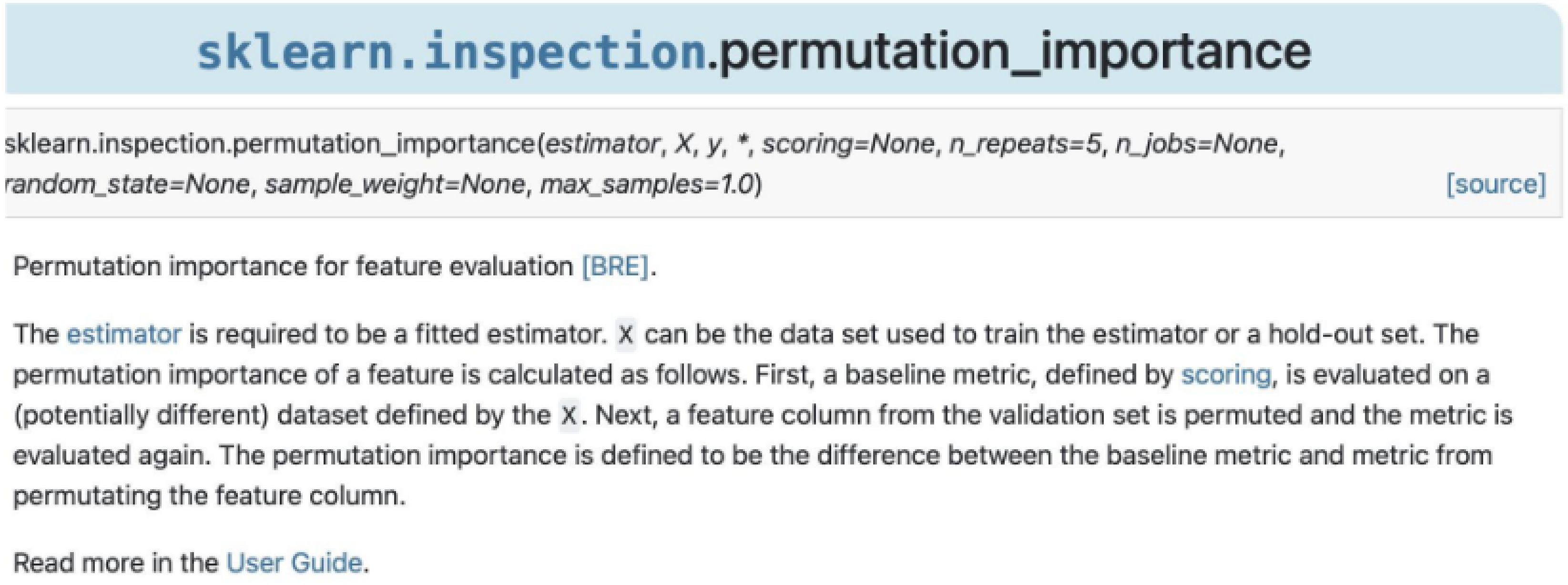
Get feature importance of each feature. For tree model Importance type can be defined as:

- 'weight': the number of times a feature is used to split the data across all trees.
- 'gain': the average gain across all splits the feature is used in.
- 'cover': the average coverage across all splits the feature is used in.
- 'total_gain': the total gain across all splits the feature is used in.
- 'total_cover': the total coverage across all splits the feature is used in.



Permutation based feature importance

- Randomly shuffle each feature and compare the model's performance change. The most important feature impact the performance the most.
- Well-supported in sklearn
- A bit slow!



sklearn.inspection.permutation_importance

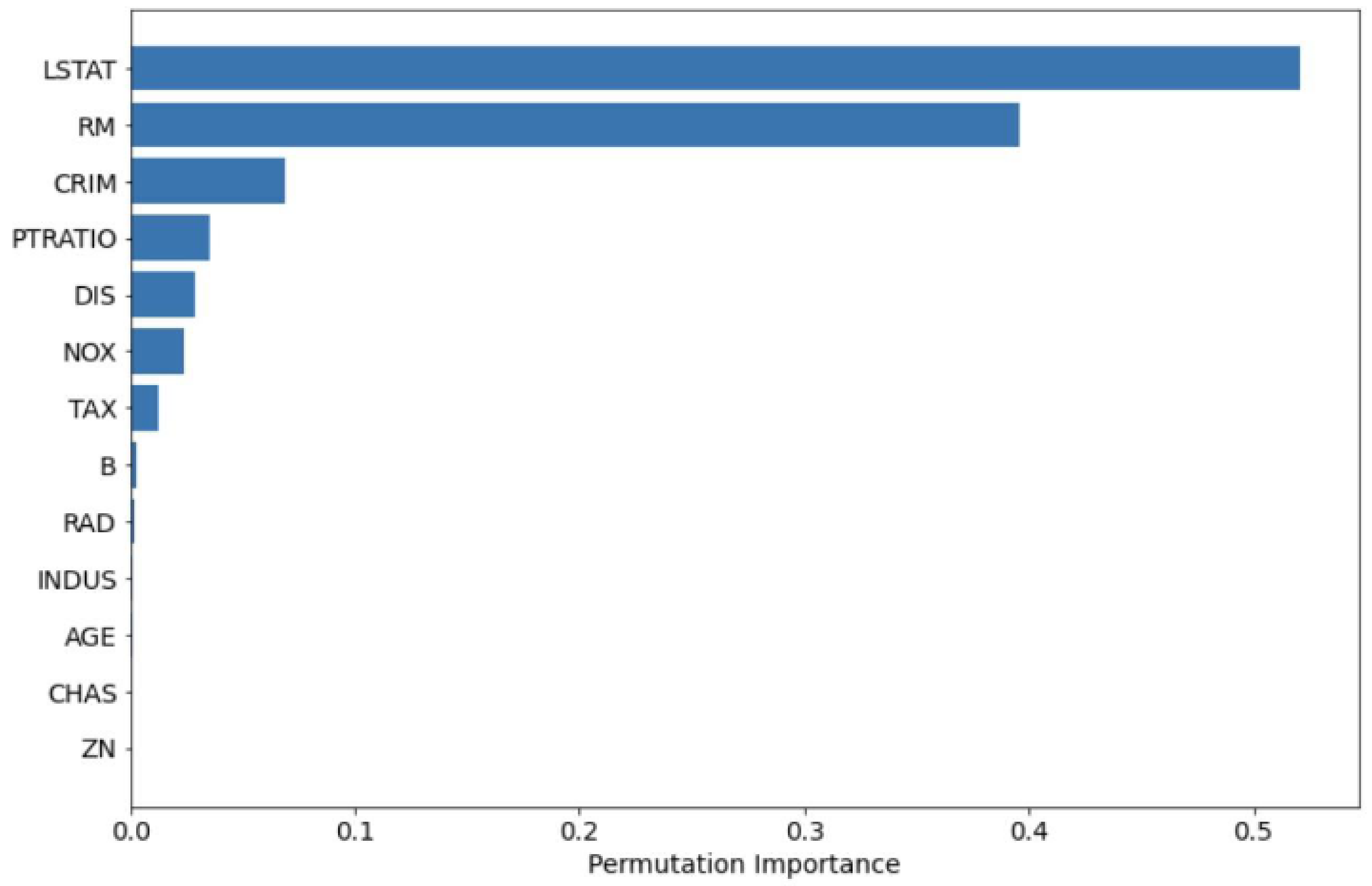
sklearn.inspection.permutation_importance(*estimator, X, y, *, scoring=None, n_repeats=5, n_jobs=None, random_state=None, sample_weight=None, max_samples=1.0*) [source]

Permutation importance for feature evaluation [BRE].

The estimator is required to be a fitted estimator. `X` can be the data set used to train the estimator or a hold-out set. The permutation importance of a feature is calculated as follows. First, a baseline metric, defined by scoring, is evaluated on a (potentially different) dataset defined by the `X`. Next, a feature column from the validation set is permuted and the metric is evaluated again. The permutation importance is defined to be the difference between the baseline metric and metric from permutating the feature column.

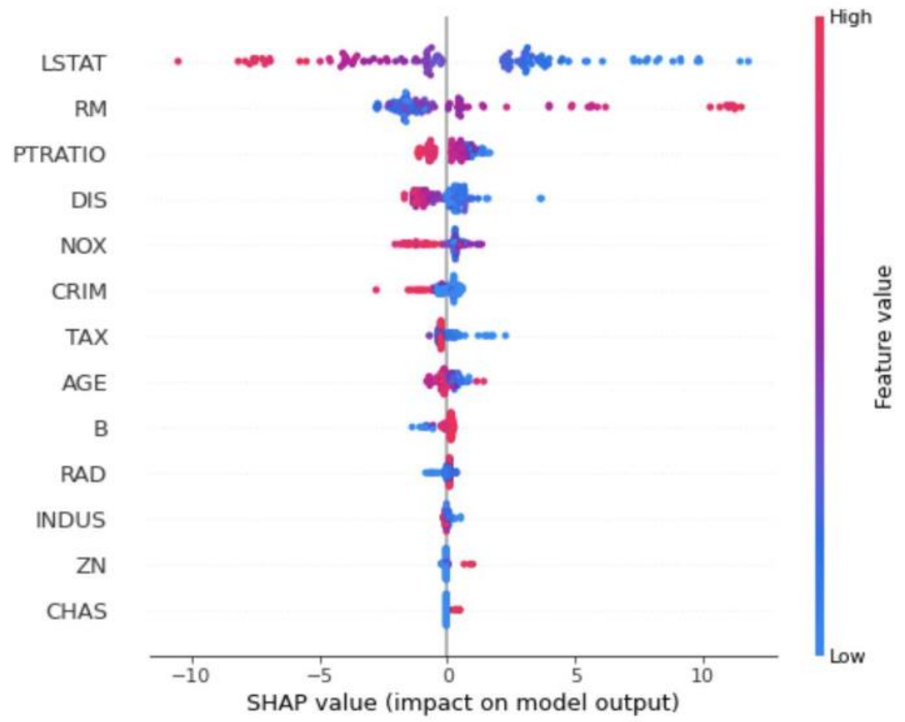
Read more in the User Guide.

Permutation importance

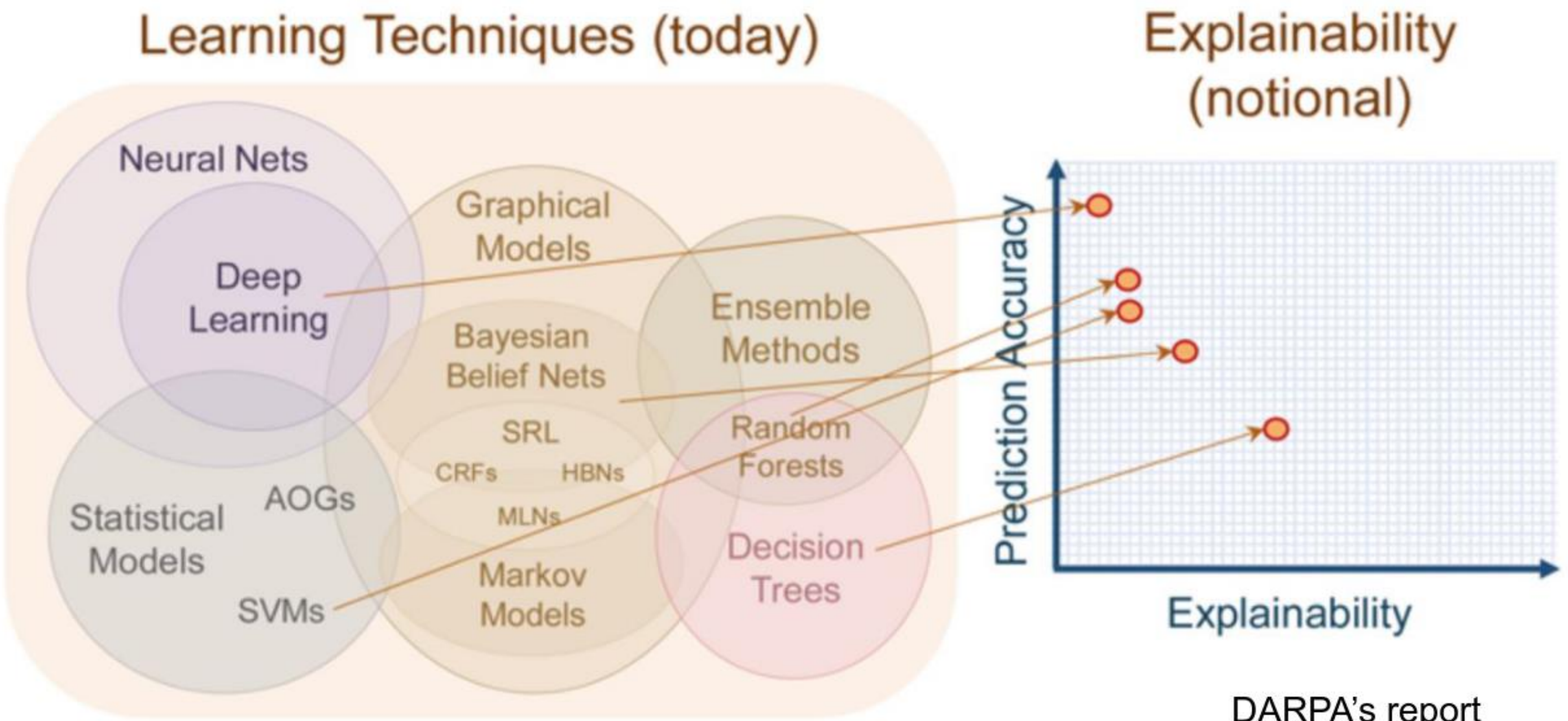


SHAP: SHapley Additive exPlanations

- Similar to LIME, it is also model-agnostic and compute the shapley value based on game theory to measure the contribution from each feature.
- Measuring the feature importance to the entire model

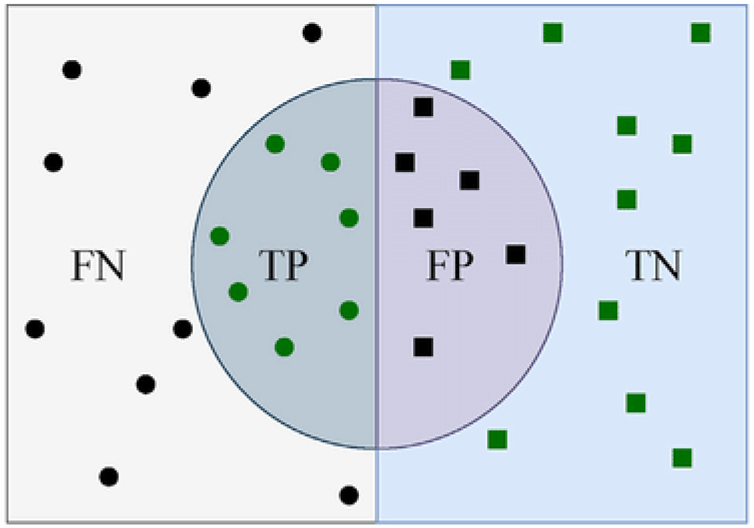


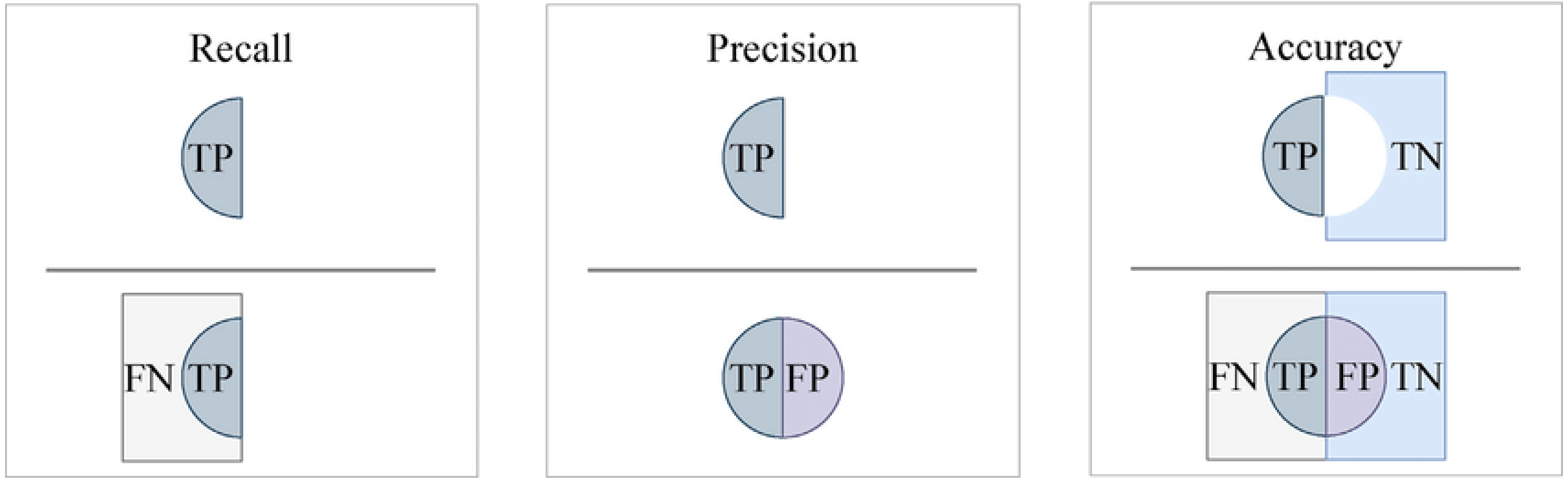
Example-driven



DARPA's report

Summary


FN
TP
FP
TN


Recall
TP
FN TP
Precision
TP
TP FP
Accuracy
TP TN
FN TP FP TN

2 Responsible M

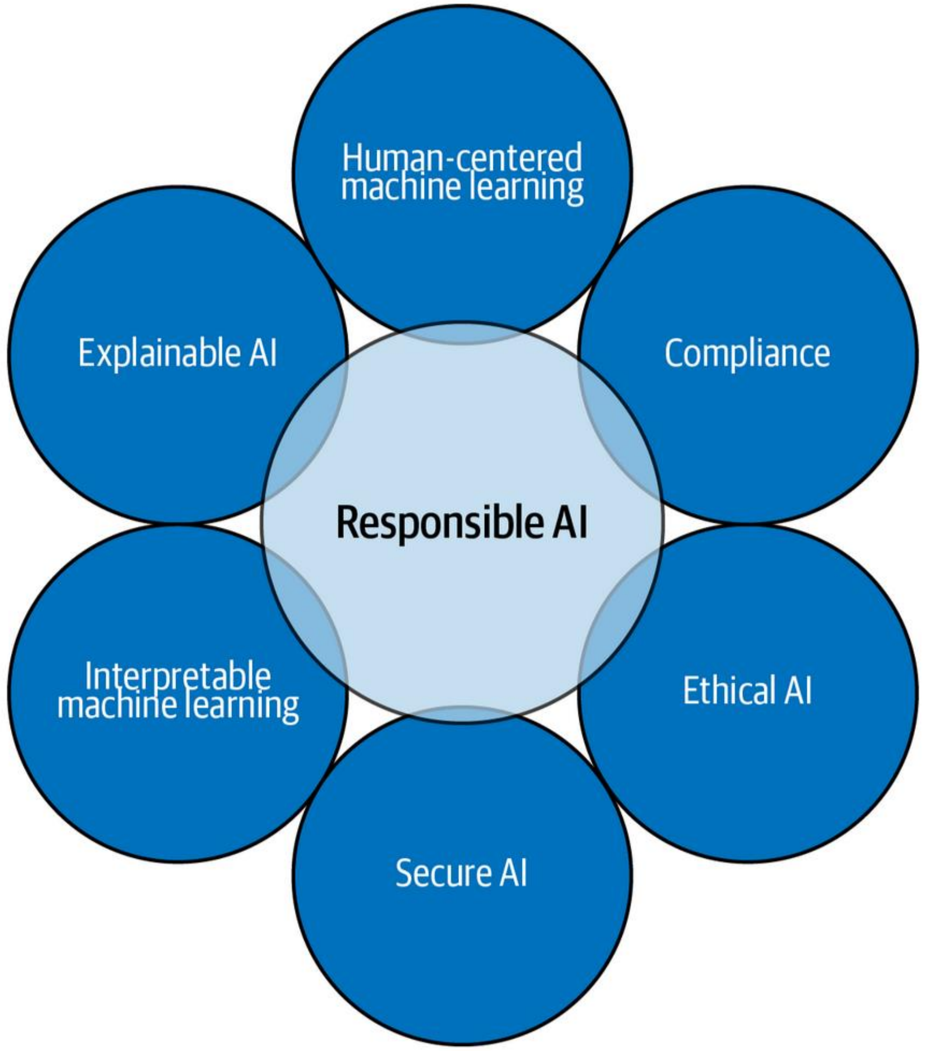
Introduction to Responsible Machine Learning

Responsible Artificial Intelligence is about human responsibility for the development of intelligent systems along fundamental human principles and values, to ensure human-flourishing and well-being in a sustainable world.

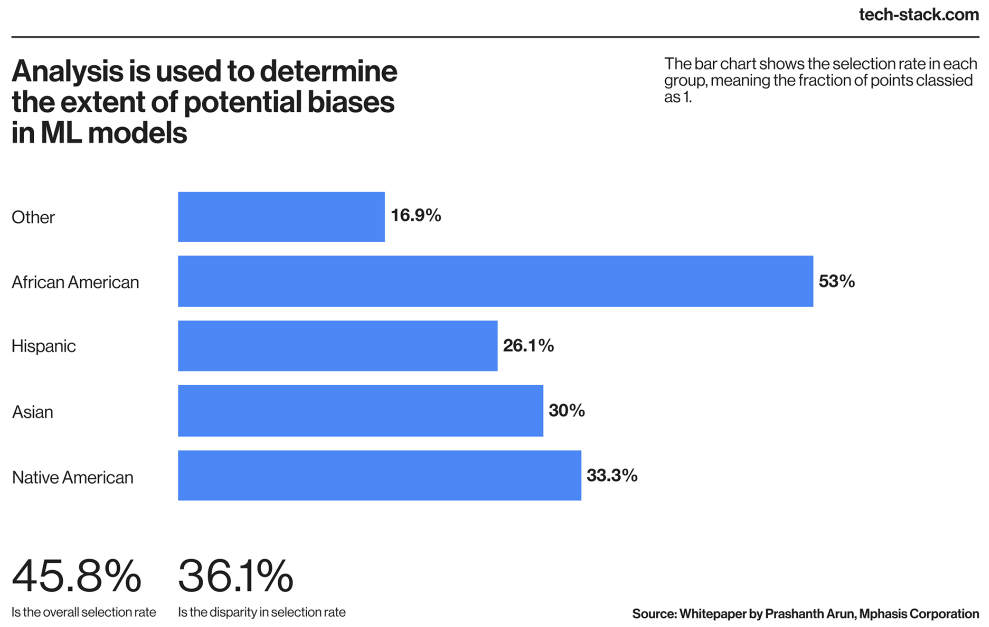
- Definition: The practice of developing and using machine learning (ML) algorithms in a way that empowers humans while minimizing negative impacts.
- Relevance: ML and AI are increasingly integrated into products we use daily, affecting user experience and societal outcomes.
- Objective: Ensure ethical, compliant, secure, and human-centered AI that benefits users and society.

Importance of Responsible ML

- User Impact:
  - Risk of offensive or discriminatory outputs without responsible practices.
  - Potential for privacy breaches and poor user experiences.
- Business Impact:
  - Harm to company reputation and customer trust.
  - Legal and ethical consequences from irresponsible ML use.
- Societal Impact:
  - Perpetuation of biases and discrimination.
  - Influence on social dynamics and equity.


Human-centered machine learning
Explainable AI
Compliance
Responsible AI
Interpretable machine learning
Ethical AI
Secure AI

Principles of Responsible Machine Learning



1. Human augmentation. The belief that ML can offer incorrect predictions, which is why it always needs humans to supervise it.
2. Bias evaluation. The commitment to continuously analyze potential biases in ML to correct them.
3. Explainability by justification. Anyone developing ML-based tools should aim to improve their transparency.
4. Reproducible operations. ML should have the proper infrastructure to guarantee reproducibility across the operations of ML systems.

Principles of Responsible Machine Learning

5. Displacement strategies. ML development should mitigate the human impact of ML adoption, especially when automation solutions displace workers.
6. Practical accuracy. ML solutions should be as precise as possible, which can only be achieved through high-quality processes.
7. Trust by privacy. The commitment to build processes that protect the data handled by ML and guarantee its privacy.
8. Data risk awareness. The belief that ML is vulnerable to attacks, which is why engineers have to constantly develop new processes to ensure a high level of security.

Incorporating Responsible ML

Selecting a Dataset Responsibly:

- Use diverse and representative data.
- Verify data sources and obtain necessary permissions.
- Ensure data is anonymized to protect privacy.

Applying Basic ML Methods Responsibly:

- Choose algorithms that are interpretable and explainable.
- Implement bias detection and mitigation techniques.
- Maintain human oversight throughout the ML process.

Extracting Conclusions Ethically:

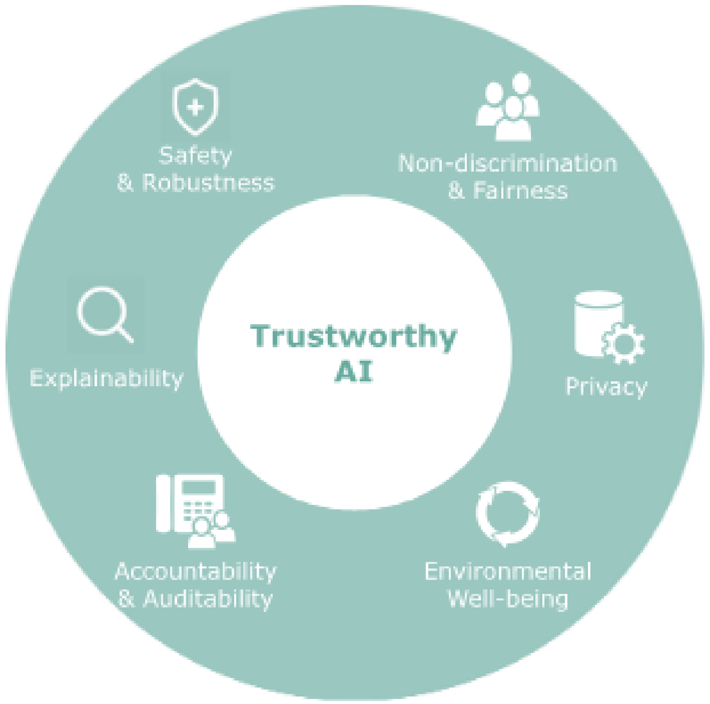
- Interpret results within the proper context.
- Be transparent about methodologies and limitations.
- Consider the societal impact of your conclusions.

3 Trustworthy ML

Trustworthy Artificial Intelligence (AI) defining it as programs and systems designed to solve problems like humans, providing benefits and convenience without posing threats or risks of harm.

3 Perspectives:

1. Technical Perspective: Trustworthy AI should exhibit accuracy, robustness, and explainability.
2. User Perspective: AI should possess availability, usability, safety, privacy, and autonomy.
3. Social Perspective: Trustworthy AI should be law-abiding, ethical, fair, accountable, and environmentally friendly.



Technical Perspective

- Accuracy: AI systems should produce outputs that are as consistent with the ground truth as possible.
- Robustness: They should be resilient to changes and perturbations in complex and volatile real-world environments.
- Explainability: AI should be transparent and allow human understanding and analysis to minimize potential risks.

User Perspective:

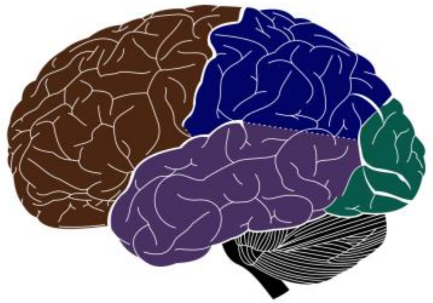
- Availability: AI systems should be accessible whenever users need them.
- Usability: They should be easy to use for people with different backgrounds.
- Safety: AI should avoid causing harm under any conditions, prioritizing user safety.
- Privacy: AI must protect user privacy and handle data responsibly.
- Autonomy: The decision-making power of AI should always be under human control.

Social Perspective

- Law-abiding and Ethical: AI should comply with all relevant laws and ethical principles.
- Fairness: AI should be non-discriminatory and ensure justice among all users.
- Accountability: There should be clear responsibility for each part of the AI system.
- Environmentally Friendly: AI should minimize energy consumption and pollution for sustainable development.

4 Neural Network

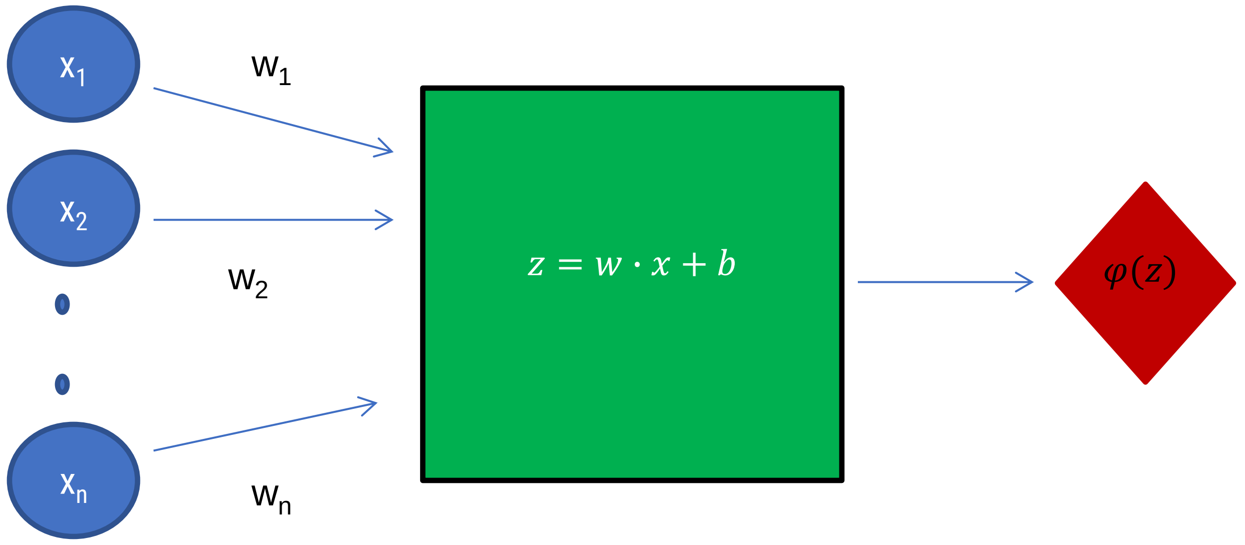
Modelling the human brain



- Artificial neural networks simulate the mode of neural interaction within the human brain, directed towards learning:
  - Base units – neurons
  - Connections between them – synapses

- The intensities (weights) of the synapses determine the performance of learning.

Artificial neuron McCulloch-Pitts



x – input vector

w – synaptic weights

b – bias

- z – linear combination unit
- φ – activation function of the neuron
- Output $\varphi(z)$

Structure of a neural network

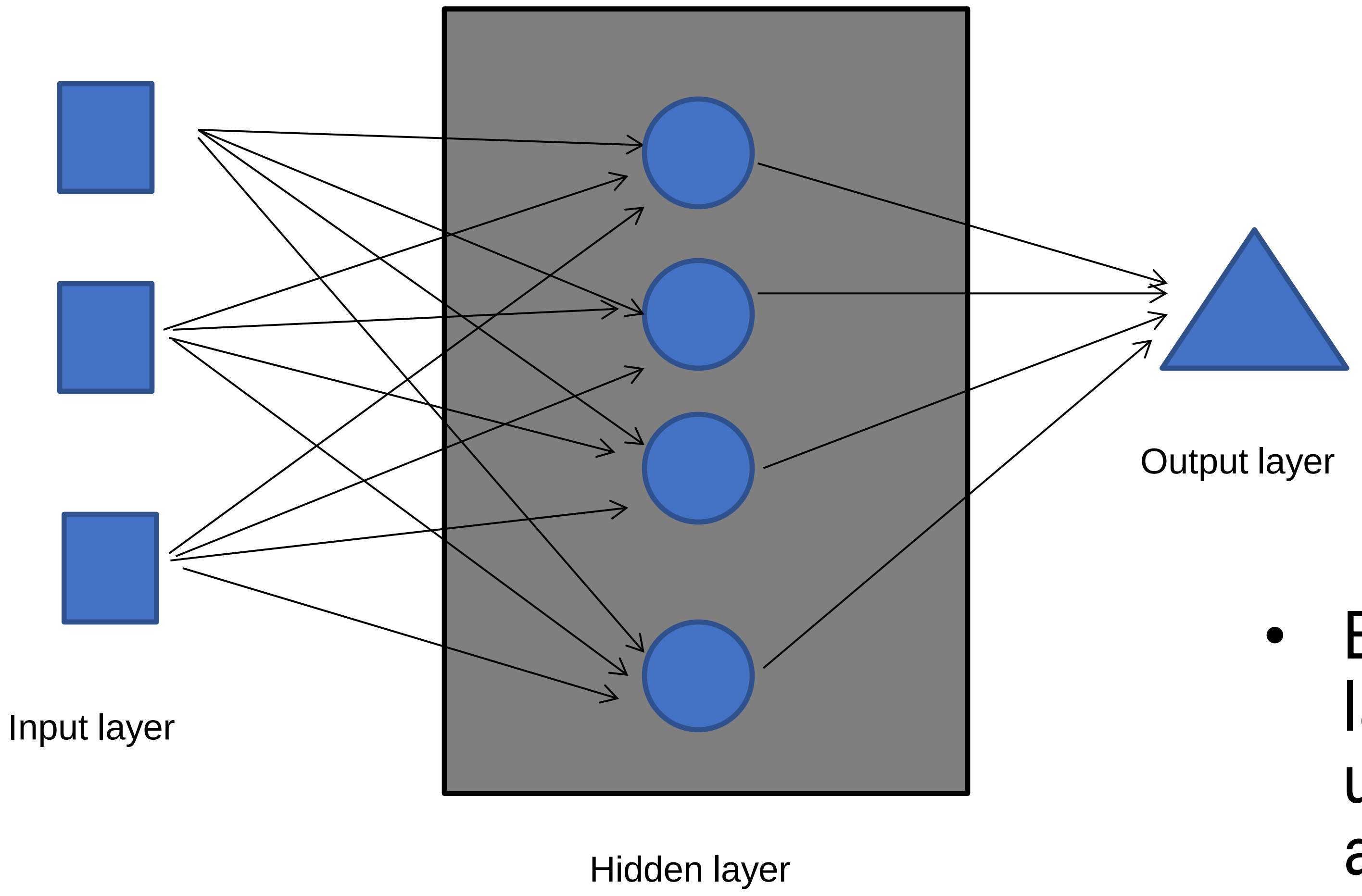
- Input neurons (units) – input layer (input data attributes)

- Hidden neurons in the black-box of learning (inside data components to be learnt)
  - In one or more hidden layers
  - The output of a layer becomes the input for the next layer

- Output neurons – output layer (network output - classes/response)

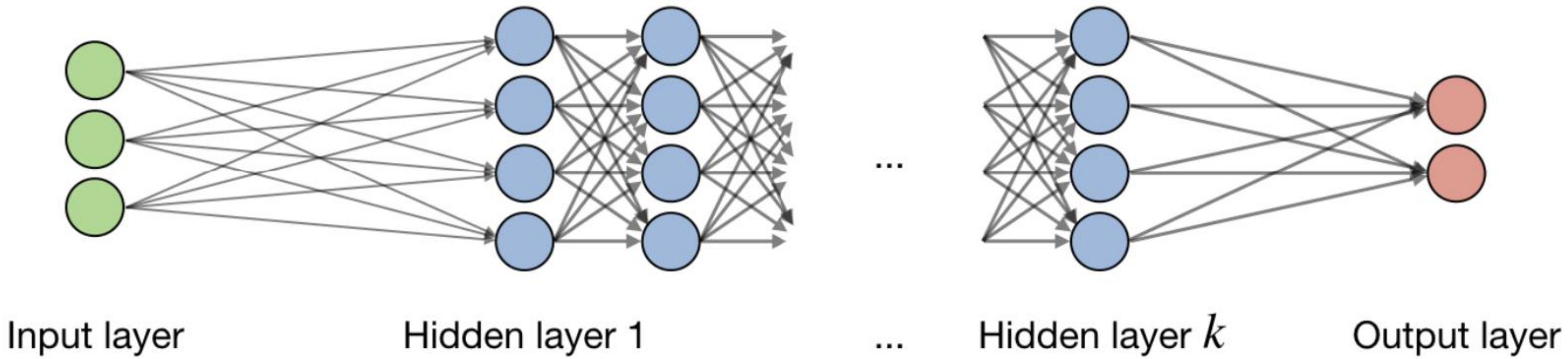
- The (supervised) learning starts from the problem data and optimizes the weights of the neurons on the base of the difference between the predicted and the real response.

Simple NN architecture



- Example with 1 hidden layer, 3 input neurons, 4 units in the hidden layer and 1 output

General NN architecture



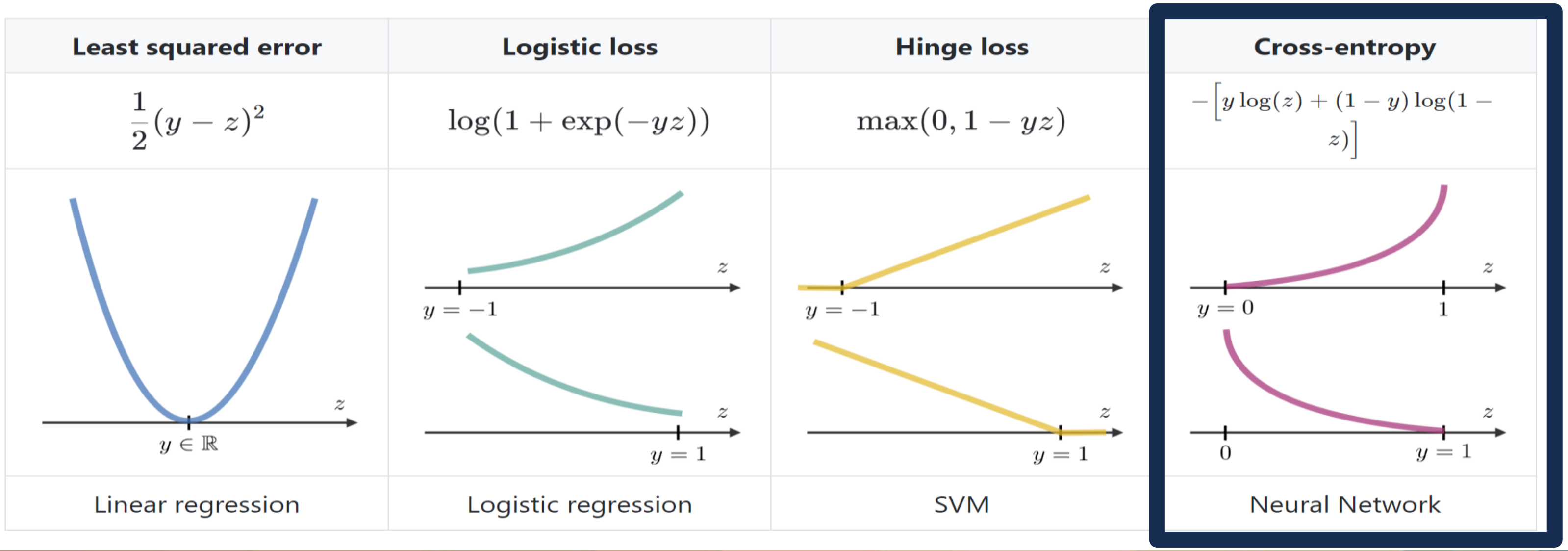
https://stanford.edu/~shervine/teaching/cs-229

NN Flow

- Batches of input data are given in the input layer
  - Batch size – amount of data passed in one forward pass iteration
- Neurons are multiplied by their weights
  - The product is further transformed by the activation function
  - The new collection of neurons are fed to the next layer
- The output of the last layer – the prediction – is compared to the output -> the loss -> to be minimized
  - Weights are initially randomly initialized
  - But, with the loss, the gradient of the cost function is calculated for the weights
  - Weights are updated after each iteration
- All data batches pass – one epoch ends
- The process is repeated for a number of epochs

Loss function

- The difference between the predicted value of the model z and the corresponding data real output y
- Cost function – sum of loss over all training data (or batch)
- NN use (binary, categorical) cross-entropy loss since it allows probability estimates; MSE for regression



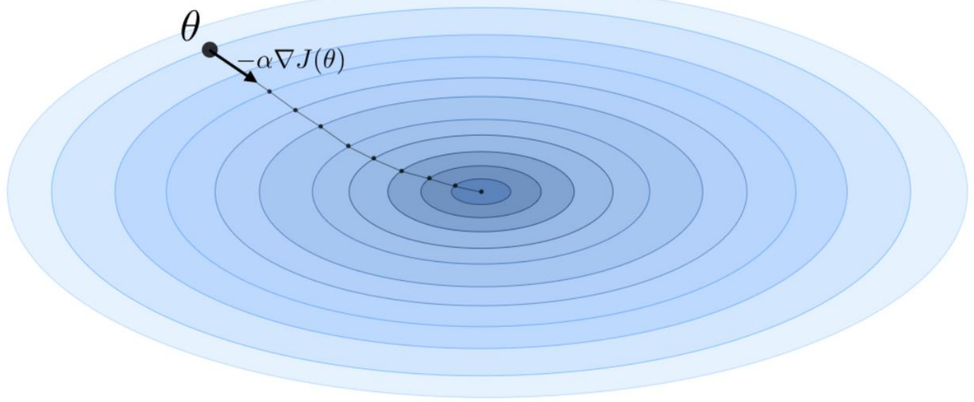
Least squared error	Logistic loss	Hinge loss	Cross-entropy
$\frac{1}{2}(y-z)^2$	$\log(1+\exp(-yz))$	$\max(0,1-yz)$	$-\left[y\log(z)+(1-y)\log(1-z)\right]$
Linear regression	Logistic regression	SVM	Neural Network

z
$y \in \mathbb{R}$
$y=-1$
$y=1$
$y=0$
0
1

https://stanford.edu/~shervine/teaching/cs-229

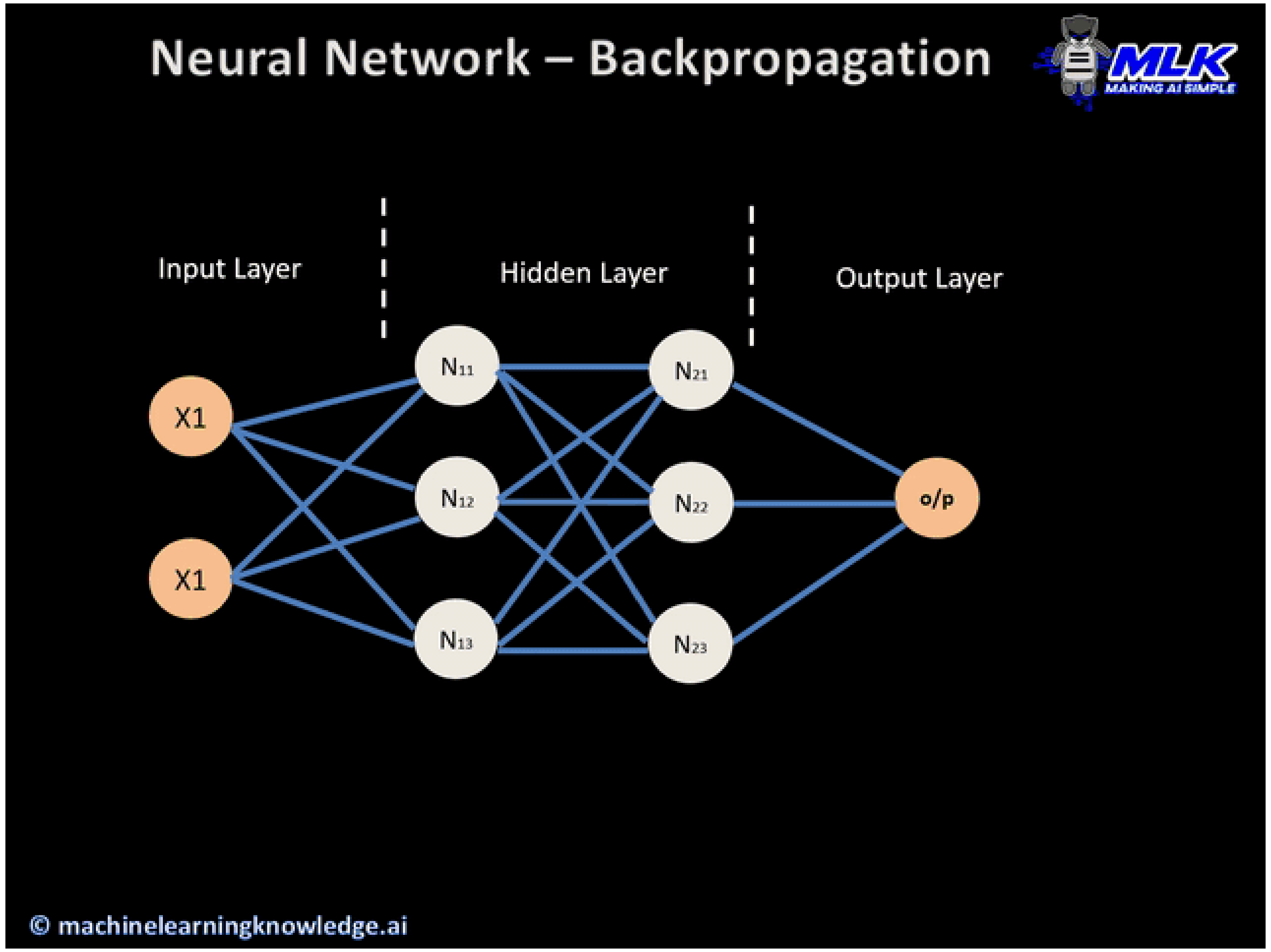
Backpropagation

- **Backpropagation**
  - The process of propagating the error backward in order to update the weights
- **Gradient descent**
  - Optimization algorithm to find the best weight values based on the gradient
- **Learning rate is the rate of updating the weights**
  - Constant
  - Decreasing gradually by time step
  - Adaptive (e.g. for the Adam optimizer)



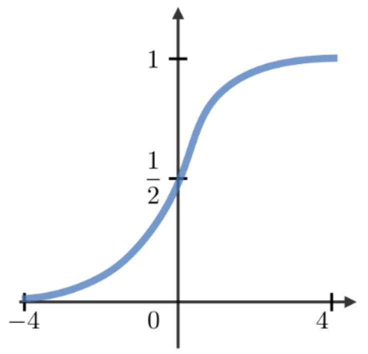
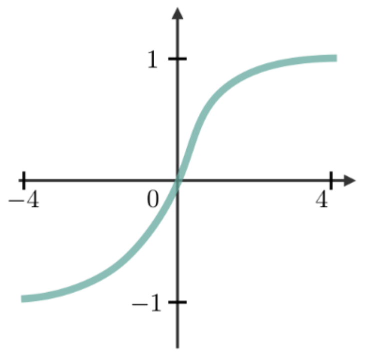
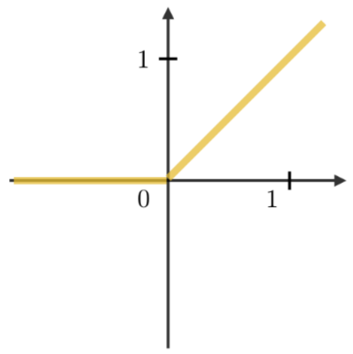
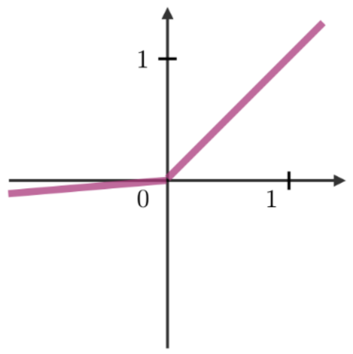
https://stanford.edu/~shervine/teaching/cs-229

Illustration



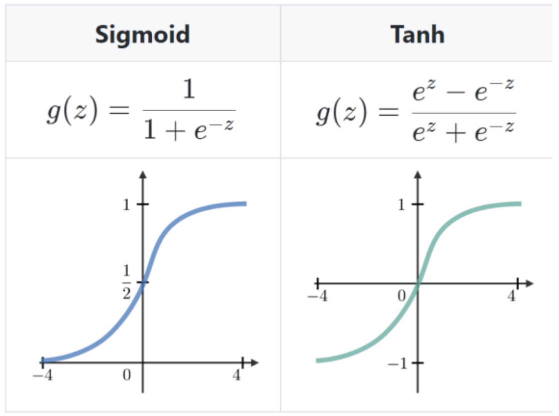
Activation function

- It induces non-linearity to the computed unit z.

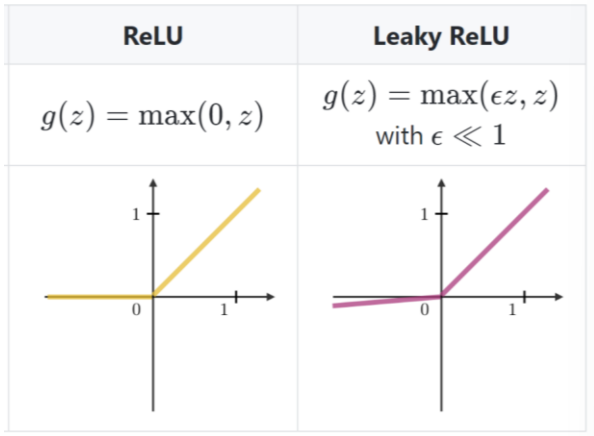
Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \dfrac{1}{1+e^{-z}}$	$g(z) = \dfrac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$
			

Choice of activation

- Sigmoid (logistic): smooth gradient; normalized output (between 0 and 1); clear prediction
  - Softmax – the extension to multi-class; used usually for last layer
  - TanH: variant for zero centered
- ReLU: computationally faster
  - But *dying ReLU* for negative or zero input (i.e., gradient is 0 -> no learning)
  - Leaky ReLU: variant for negative or zero input to still have backpropagation



Sigmoid	Tanh
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$



ReLU	Leaky ReLU
$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$

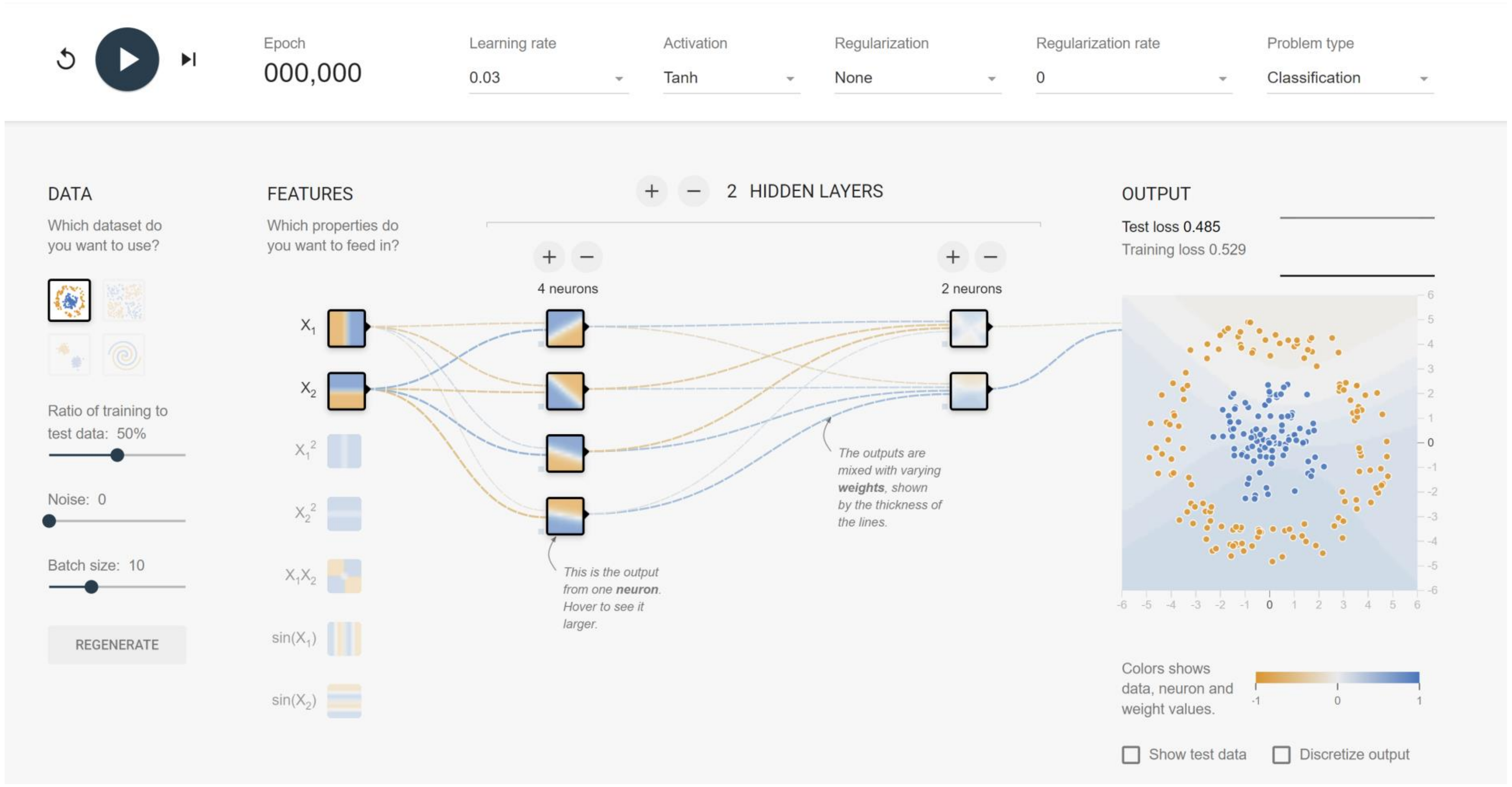
Regularization

- Assumption: a model with smaller weights is less complex than one with larger weights

- Penalization for the weights added to the cost function
- λ - regularization rate
- L1 norm is the sum of absolute values for the weights $\lambda \sum |w_i|$
- L2 norm is the sum of squared values for the weights $\lambda \sum w_i^2$

Example of NN flow

http://playground.tensorflow.org/


Epoch 000,000

Learning rate 0.03

Activation Tanh

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

2 HIDDEN LAYERS

4 neurons

2 neurons

*This is the output from one **neuron**. Hover to see it larger.*

*The outputs are mixed with varying **weights**, shown by the thickness of the lines.*

OUTPUT

Test loss 0.485

Training loss 0.529

Colors shows data, neuron and weight values.

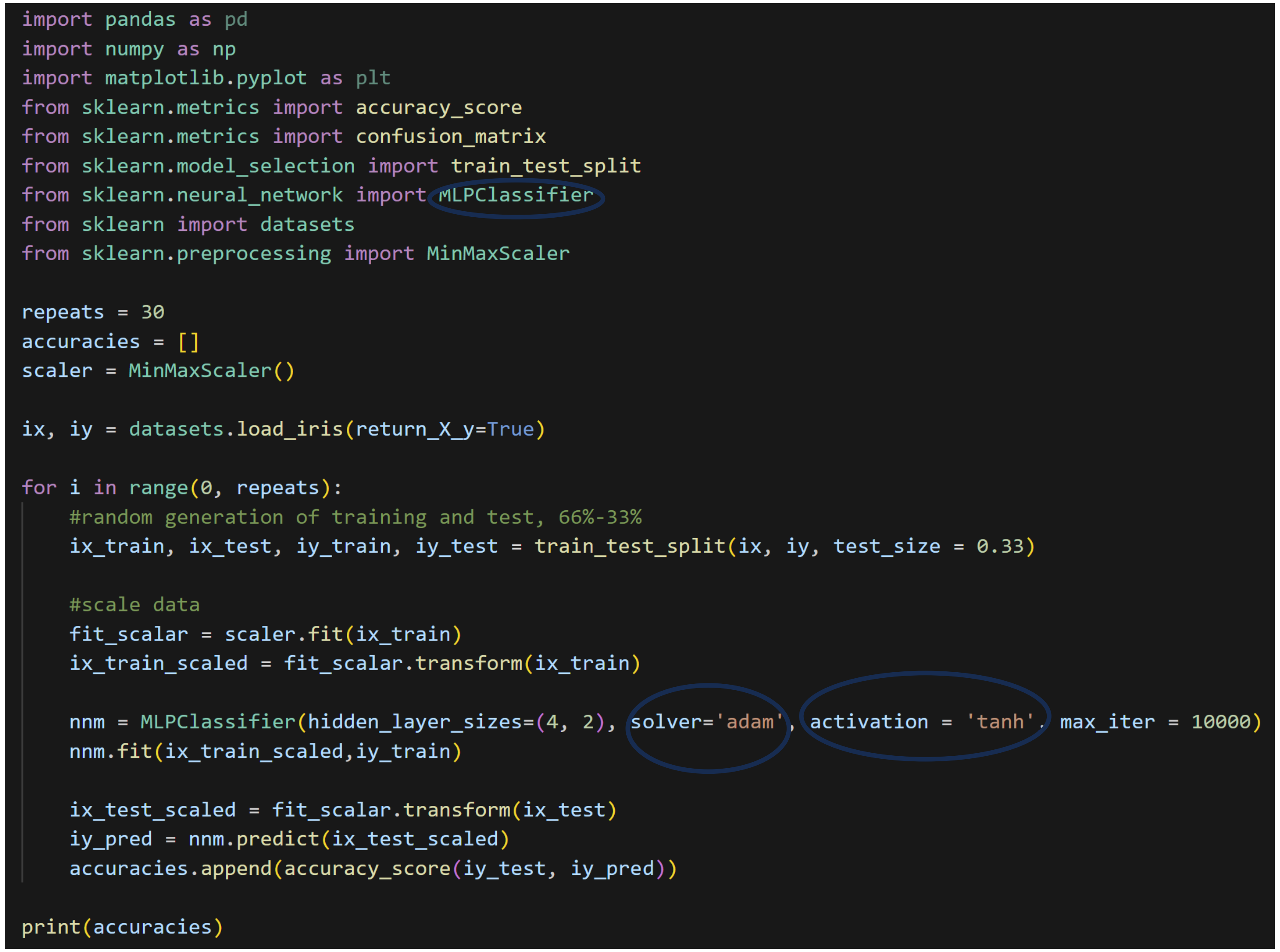
Show test data

Discretize output

NN in Python

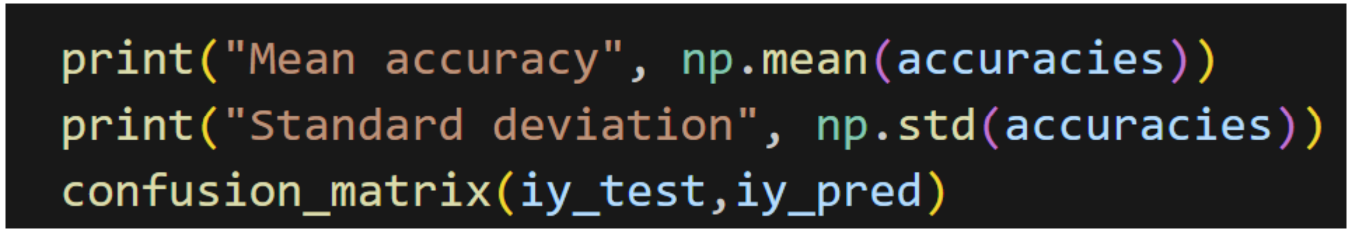
- In *sklearn* and the dedicated sublibrary *neural_network*, there is the function *MLPClassifier*.

- Gradient algorithms:
  - Stochastic gradient descent (SGD)
  - Adam

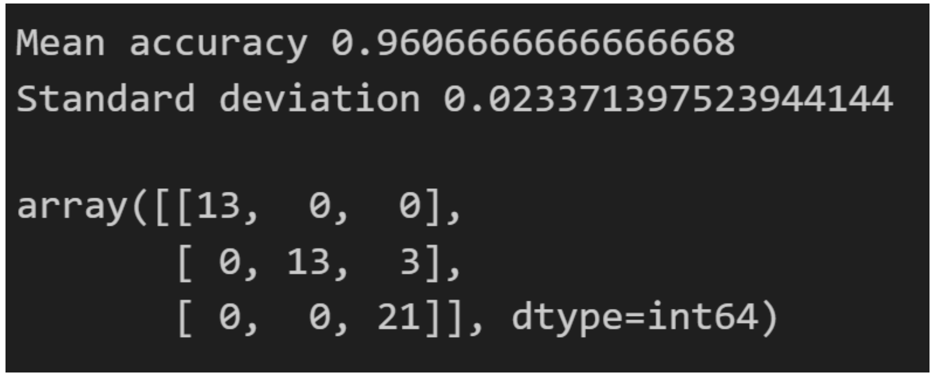


```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn import datasets
from sklearn.preprocessing import MinMaxScaler

repeats = 30
accuracies = []
scaler = MinMaxScaler()

ix, iy = datasets.load_iris(return_X_y=True)

for i in range(0, repeats):
    #random generation of training and test, 66%-33%
    ix_train, ix_test, iy_train, iy_test = train_test_split(ix, iy, test_size = 0.33)

    #scale data
    fit_scalar = scaler.fit(ix_train)
    ix_train_scaled = fit_scalar.transform(ix_train)

    nnm = MLPClassifier(hidden_layer_sizes=(4, 2), solver='adam', activation = 'tanh', max_iter = 10000)
    nnm.fit(ix_train_scaled,iy_train)

    ix_test_scaled = fit_scalar.transform(ix_test)
    iy_pred = nnm.predict(ix_test_scaled)
    accuracies.append(accuracy_score(iy_test, iy_pred))

print(accuracies)
```

Results

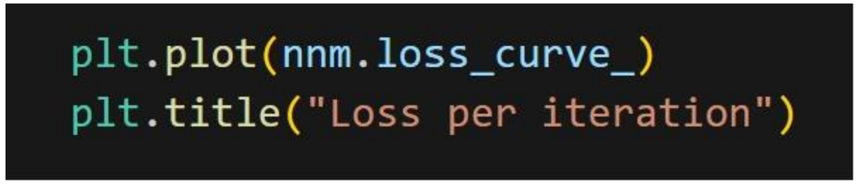


```python
print("Mean accuracy", np.mean(accuracies))
print("Standard deviation", np.std(accuracies))
confusion_matrix(iy_test,iy_pred)
```

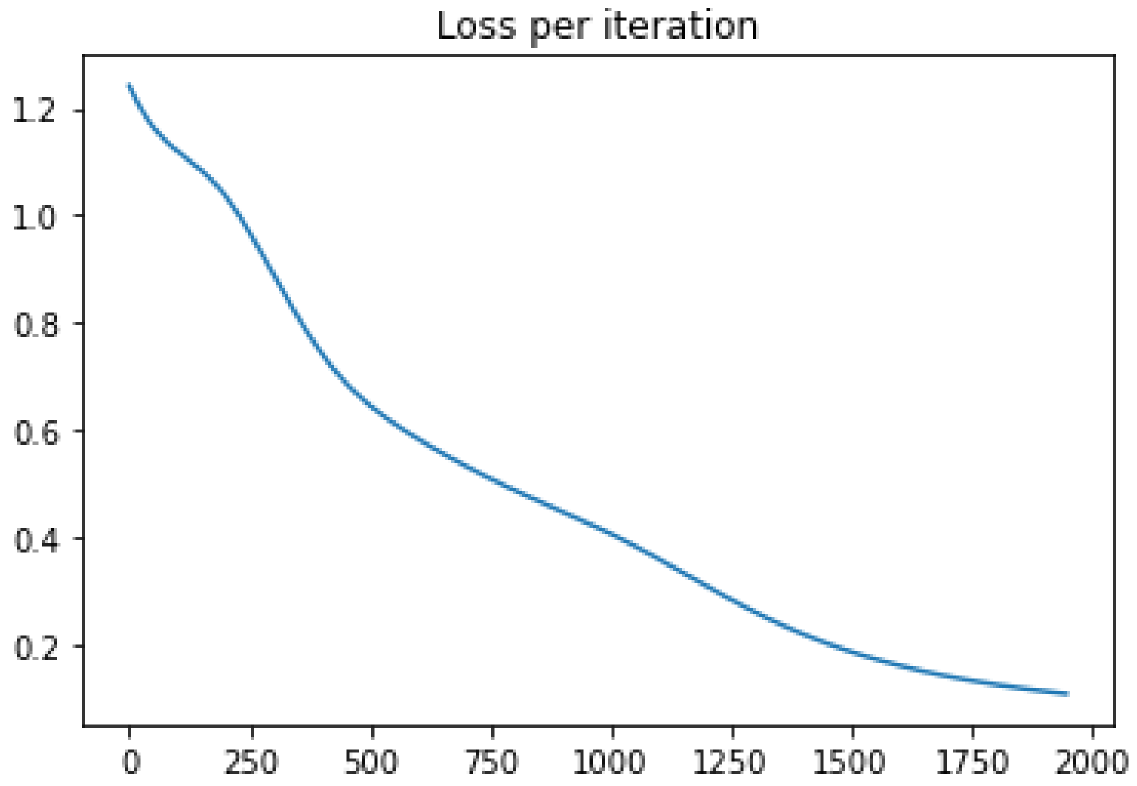


```
Mean accuracy 0.9606666666666668
Standard deviation 0.023371397523944144

array([[13,  0,  0],
       [ 0, 13,  3],
       [ 0,  0, 21]], dtype=int64)
```

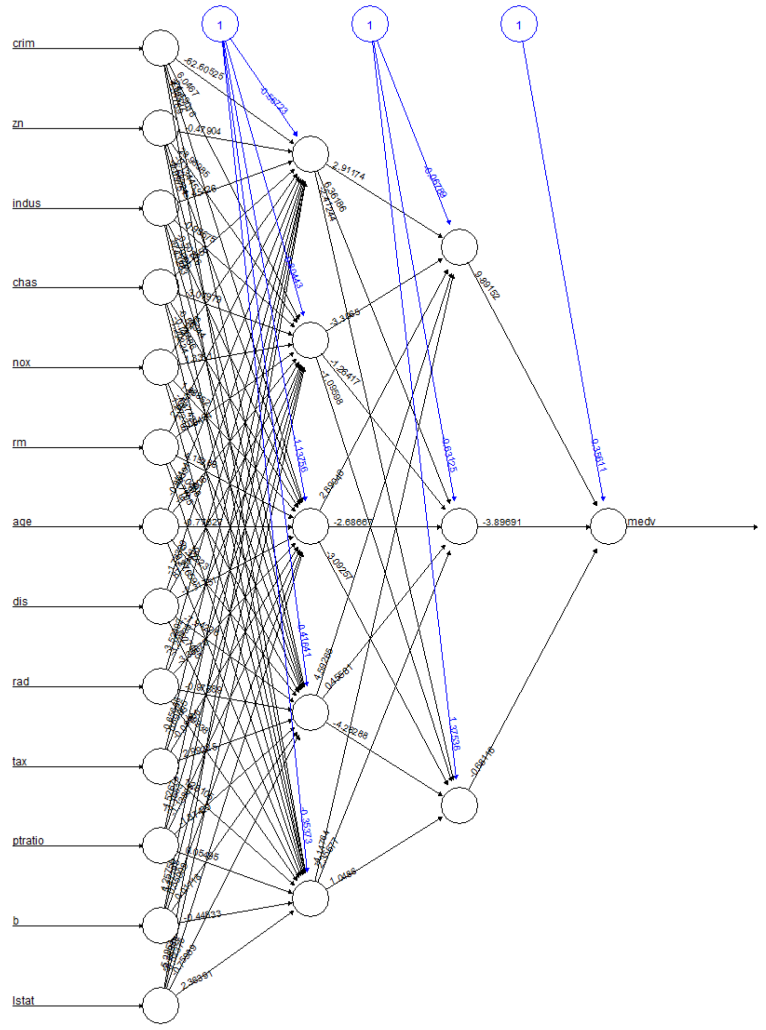
Loss decrease

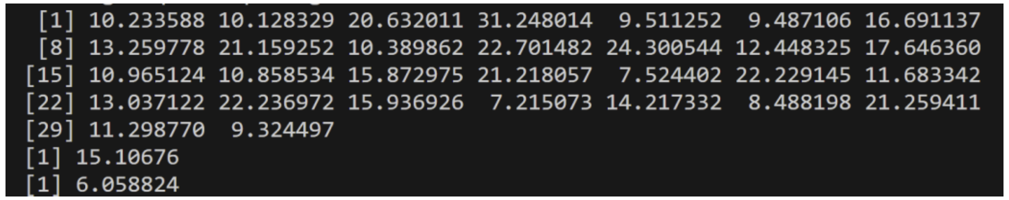


```
plt.plot(nnm.loss_curve_)
plt.title("Loss per iteration")
```

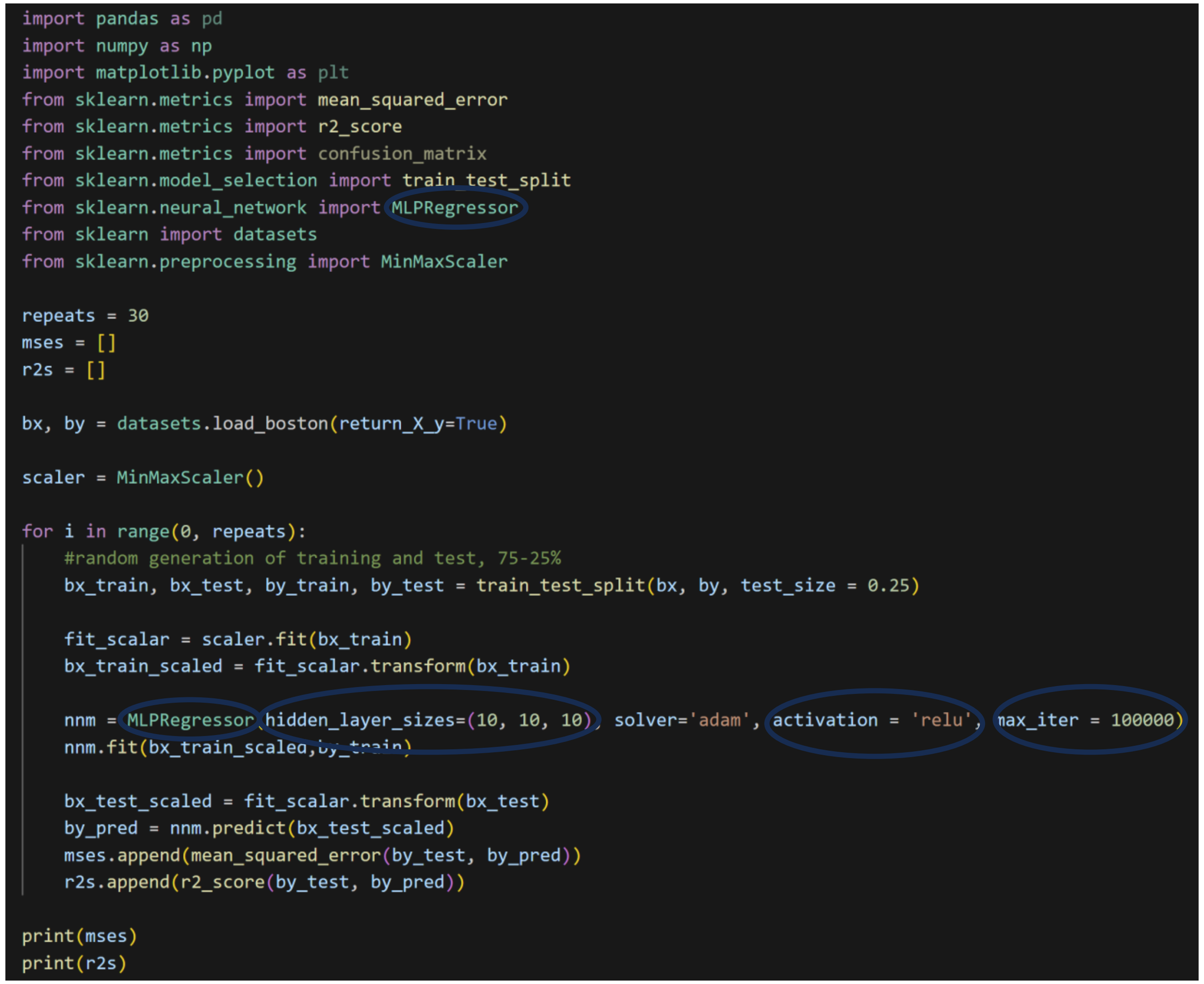


Results Boston housing




```
  [1] 10.233588 10.128329 20.632011 31.248014  9.511252  9.487106 16.691137
  [8] 13.259778 21.159252 10.389862 22.701482 24.300544 12.448325 17.646360
 [15] 10.965124 10.858534 15.872975 21.218057  7.524402 22.229145 11.683342
 [22] 13.037122 22.236972 15.936926  7.215073 14.217332  8.488198 21.259411
 [29] 11.298770  9.324497
[1] 15.10676
[1] 6.058824
```

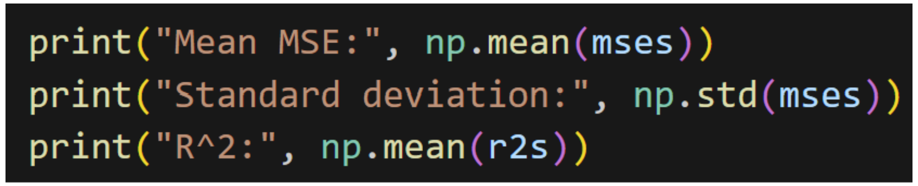
NN regression in Python



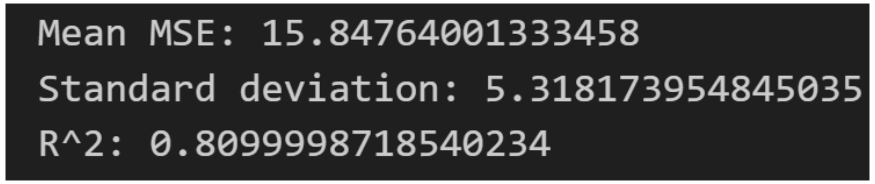
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn import datasets
from sklearn.preprocessing import MinMaxScaler

repeats = 30
mses = []
r2s = []

bx, by = datasets.load_boston(return_X_y=True)

scaler = MinMaxScaler()

for i in range(0, repeats):
    #random generation of training and test, 75-25%
    bx_train, bx_test, by_train, by_test = train_test_split(bx, by, test_size = 0.25)

    fit_scalar = scaler.fit(bx_train)
    bx_train_scaled = fit_scalar.transform(bx_train)

    nnm = MLPRegressor(hidden_layer_sizes=(10, 10, 10), solver='adam', activation = 'relu', max_iter = 100000)
    nnm.fit(bx_train_scaled,by_train)

    bx_test_scaled = fit_scalar.transform(bx_test)
    by_pred = nnm.predict(bx_test_scaled)
    mses.append(mean_squared_error(by_test, by_pred))
    r2s.append(r2_score(by_test, by_pred))

print(mses)
print(r2s)
```

Boston results

[17.492755007264936, 30.234505625912554, 10.02046071680119, 19.037853523236794, 18.448766845564116, 20.87579946196899, 8.826236048369347, 24.001715622499553, 18.878682321430265, 12.02416203522131, 22.035370209214424, 9.329690404116555, 12.149204856579493, 23.005473435020107, 8.454154531507836, 14.288911607219333, 13.012196991202496, 10.710487549511683, 17.14418010662683, 14.955289395359628, 23.199785268773645, 11.473790827049333, 11.602084873366444, 11.009107901891278, 14.504382535062557, 21.252839627988767, 10.808259854302339, 18.340272996999875, 13.727357417027171, 14.585422802948548]



```
print("Mean MSE:", np.mean(mses))
print("Standard deviation:", np.std(mses))
print("R^2:", np.mean(r2s))
```



```
Mean MSE: 15.84764001333458
Standard deviation: 5.318173954845035
R^2: 0.8099998718540234
```