Medical/Bio Research Topics Ⅰ: Week 11 (16.05.2024)

Development of Brain Imaging-based Artificial Intelligence Models (2): Model Construction

뇌영상 기반 인공지능 모델 개발 연습 (2): 예측 모델 구성

Automated Machine Learning (AutoML)

- Process of automating the end-to-end process of applying machine learning to real-world problems
- Aims to automate the time-consuming and challenging tasks involved in building machine learning models, thereby reducing the need for extensive domain expertise and allowing for more rapid experimentation and iteration

- General features
  - Data ingestion and preprocessing
    - Handles tasks such as data cleaning, formatting, and feature engineering, often automatically detecting the data types and applying appropriate transformations
  - Model selection
    - Explores different model architectures and algorithms that are suitable for the given problem and dataset
  - Hyperparameter tuning
    - Automatically tunes the hyperparameters of the selected models, which are configurations that can significantly impact model performance

- Model evaluation and validation
  - Evaluates the performance of the trained models using appropriate metrics and techniques (e.g., cross-validation or holdout sets)
- Model ensembling
  - May combine multiple models through ensembling techniques (e.g., stacking, bagging, or blending) to improve overall performance and robustness
- Model deployment
  - Deploys the best-performing model or ensemble for production use, often with automatic code generation and containerization for easy deployment

- Benefits
  - Efficiency
    - Reduces the time required to build and optimize models
  - Accessibility
    - Makes machine learning techniques accessible to users without extensive machine learning expertise
  - Performance
    - Often achieves competitive or superior performance compared to manual model tuning

- Popular AutoML libraries in Python
  - Primarily focused on automating the process of selecting traditional machine learning algorithms
    - TPOT (Tree-based Pipeline Optimization Tool) [http://epistasislab.github.io/tpot/; https://github.com/EpistasisLab/tpot]
    - Auto-sklearn [https://automl.github.io/auto-sklearn/; https://github.com/automl/auto-sklearn]
    - H2O AutoML [https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html; https://github.com/h2oai/h2o-3]
    - PyCaret [https://pycaret.org/; https://github.com/pycaret/pycaret]
    - MLBox [https://mlbox.readthedocs.io/en/latest/; https://github.com/AxeldeRomblay/MLBox]
    - AutoGluon [https://auto.gluon.ai/stable/index.html; https://github.com/autogluon/autogluon]

- Designed specifically for automating the process of neural architecture search and hyperparameter optimization for deep learning models
  - Auto-PyTorch [https://www.automl.org/automl-for-x/tabular-data/autopytorch/; https://github.com/automl/Auto-PyTorch]
  - AutoKeras [https://autokeras.com/; https://github.com/keras-team/autokeras]
- Cloud-based services
  - Google Cloud AutoML [https://cloud.google.com/automl/]
    - AutoML Vision, AutoML Video Intelligenc, AutoML Natural Language, and AutoML Tables
  - Microsoft Azure AutoML [https://azure.microsoft.com/products/machine-learning/automatedml/]
    - AutoML for Image Tasks, AutoML for Text Task, and AutoML for Forecasting Tasks

- Limitations
  - Does not entirely replace the need for human oversight and expertise
  - Still requires input data, problem formulation, and guidance from domain experts to ensure appropriate model selection, interpretation, and deployment

TPOT

- AutoML library in Python that uses a genetic algorithm (GA) to explore the space of possible machine learning pipelines and optimize them
  - Starts with a population of random pipelines and iteratively evolves them through operations like mutation (modifying pipeline steps) and crossover (combining pipelines) to find the best-performing pipeline for the given dataset
  - Automatically constructs complex machine learning pipelines that may be difficult for human experts to design manually

- GA
  - Search heuristic inspired by Charles Darwin’s theory of natural selection
    - Reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation
  - GA in TPOT
    - TPOT will evaluate POPULATION_SIZE (number of individuals to retain in the GP population every generation) + GENERATIONS (number of iterations to run the pipeline optimization process) x OFFSPRING_SIZE (= POPULATION_SIZE by default) pipelines in total
    - With the default TPOT settings (100 generations with 100 population size), TPOT will evaluate 10,000 pipeline configurations before finishing

- Benefits
  - Automates the process of testing numerous combinations of preprocessing steps and model configurations, which can be computationally expensive and tedious to do manually
  - Capable of performing a global search and less likely to be trapped in local optimum solutions compared to other optimization methods
  - Can adapt to the specific characteristics of the data by evolving pipelines that are better suited to the problem

- Challenges
  - Computationally expensive especially for large datasets or complex problems due to the need of evaluating a large number of pipelines during the optimization process
  - Stochastic nature such that runs can be somewhat different unless the random state is fixed

- Example AutoML implementation with TPOT

```
from tpot import TPOTClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Load a sample dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Initialize the TPOT classifier
tpot = TPOTClassifier(generations=5, population_size=20)

# Fit and evaluate the model
tpot.fit(X_train, y_train)
acc = tpot.score(X_test, y_test)

# Export the best pipeline
pot.export('tpot_best_pipeline.py')
```

Feature Engineering

- Crucial step in the machine learning pipeline that involves transforming raw data into meaningful features that improve the performance of machine learning models
- Blend of art and science, combining statistical techniques with domain expertise to unlock the full potential of machine learning models
  - Well-engineered features can lead to simpler, more interpretable models with better performance

- Main operations
  - Feature selection
    - Identifis and retains the most relevant features
  - Feature extraction
    - Creates new features from existing ones
  - Feature transformation
    - Modifies features to better represent the data
  - Feature interaction
    - Captures relationships between features

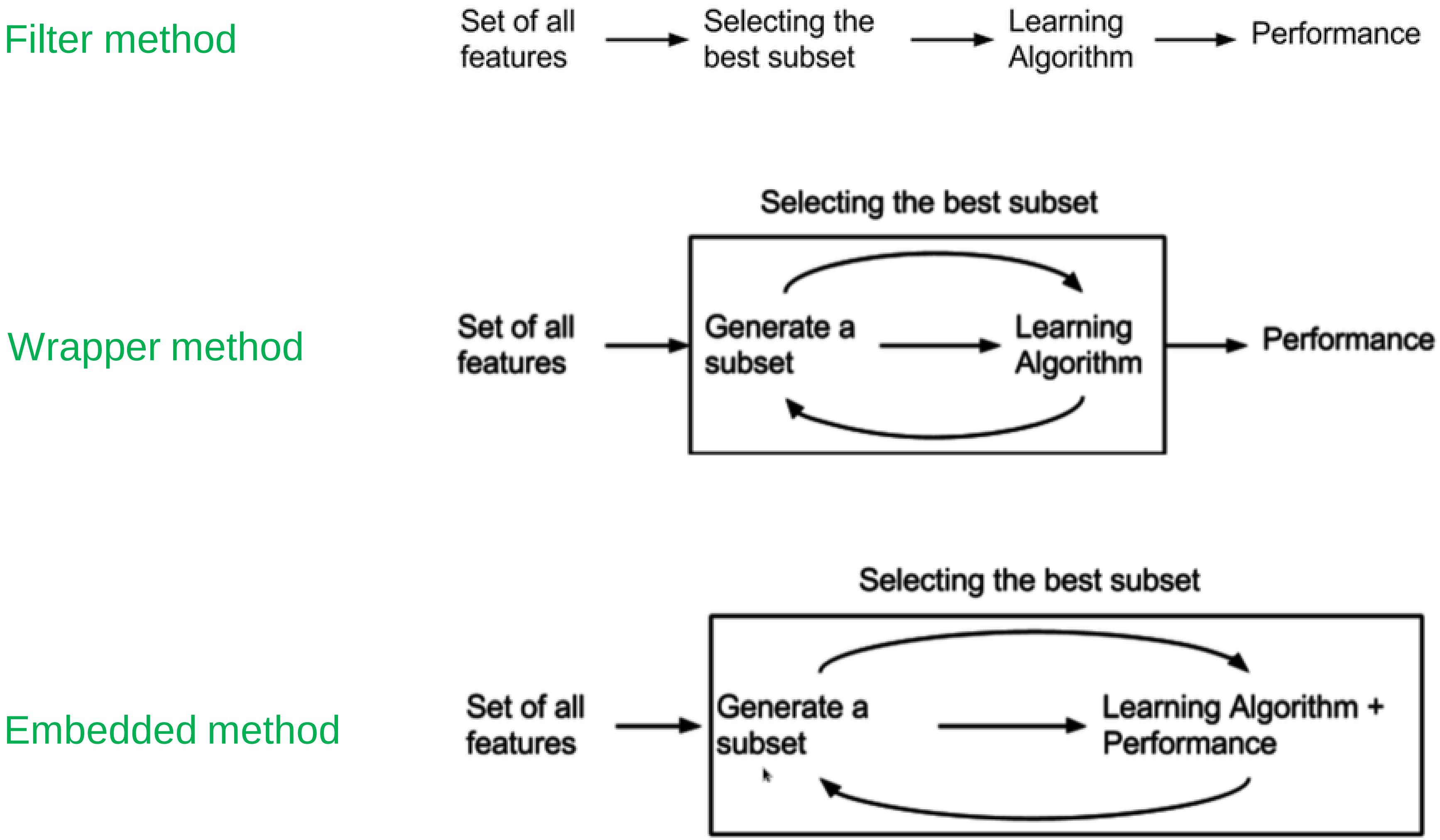
- Handling missing values
    - Addresses gaps in data
- Encoding categorical variables
    - Converts categorical data into numerical form
- Dimensionality reduction
    - Reduces the number of features while retaining important information

- Automated tools
  - Feature-engine [https://feature-engine.trainindata.com/; https://github.com/feature-engine/feature_engine]
  - Scikit-learn [https://scikit-learn.org/; https://github.com/scikit-learn/scikit-learn]
  - tsfresh [https://tsfresh.readthedocs.io/; https://github.com/blue-yonder/tsfresh]

Feature Selection

- Methods
  - Filter method
    - Ranks and selects features based on their scores or statistical measures calculated directly from the data, such as correlation, mutual information, χ^2-statistic from the χ^2 test, or F-statistic from the ANOVA, without involving any learning algorithm
    - May fail to capture feature dependencies and interactions

- Wrapper method
  - Iteratively adds or removes features and evaluates the performance of a model using techniques such as forward selection, backward elimination, or recursive feature elimination
  - Computationally expensive, especially for high-dimensional datasets
- Embedded method
  - Performs feature selection as part of the model construction process
  - Some machine learning algorithms like LASSO, ridge regression, and decision trees have built-in mechanisms for feature selection by assigning weights or importances to features during the training process
  - Specific to the machine learning algorithm used



[https://en.wikipedia.org/wiki/Feature_selection]

Feature selection methods

- LASSO (Least Absolute Shrinkage and Selection Operator)
  - Regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces
  - Key features
    - Regularization
      - Adds a penalty term that discourages large coefficients, which helps in shrinking some coefficients to exactly zero and thus reducing overfitting
    - Feature selection
      - Automatically selects a simpler model that only includes the most relevant features by driving some coefficients to zero
    - Interpretability
      - Induces a model that is easier to interpret because it is simpler and contains only the most important features

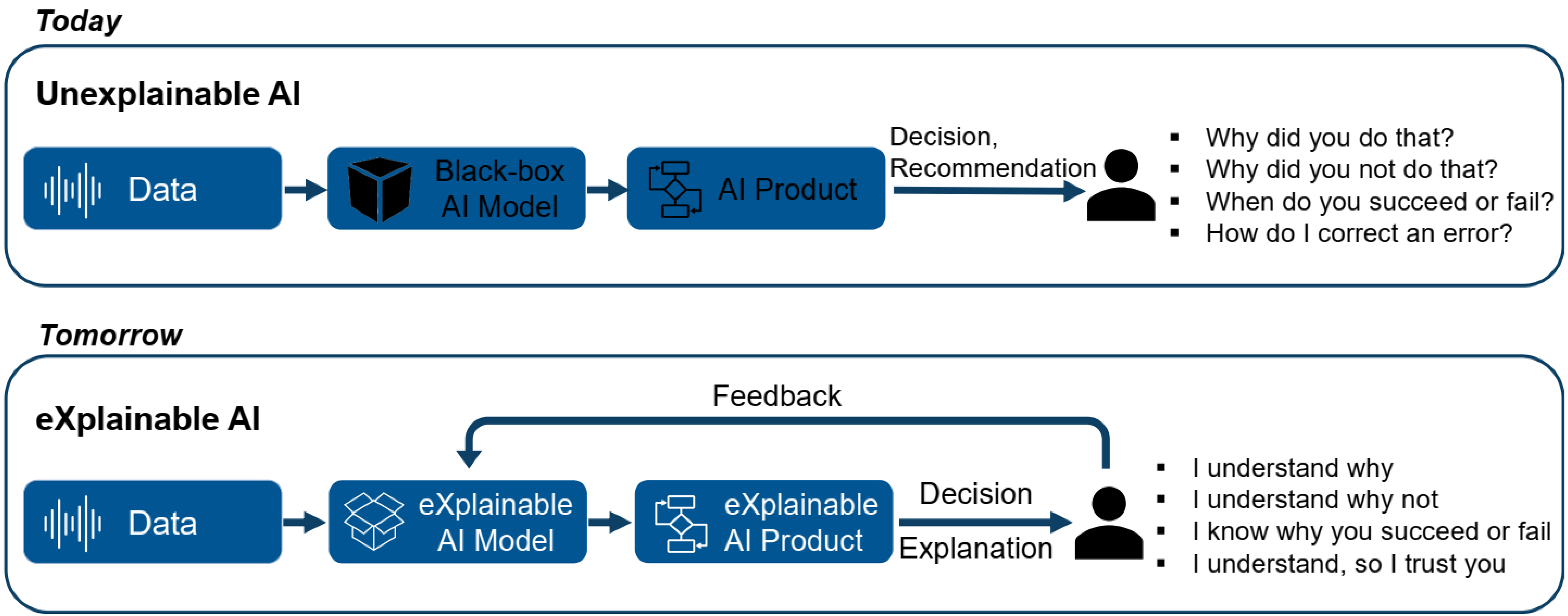
– Mathematical formulation

Objective function = $\sum_{i=1}^{n}(\mathcal{Y}_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$

- Shrinkage
  - As the regularization parameter (λ) increases, the coefficients are shrunk towards zero
- Selection
  - Some coefficients may become exactly zero, which means the corresponding features are effectively removed from the given model

Explainable Artificial Intelligence (XAI)

- Field that focuses on making artificial intelligence (AI) systems more interpretable, transparent, and understandable to humans
- Aims to provide explanations for the predictions made by AI models, particularly in high-stakes domains where accountability, fairness, and trust are crucial



Today

Unexplainable AI

Data → Black-box AI Model → AI Product → Decision, Recommendation

- Why did you do that?
- Why did you not do that?
- When do you succeed or fail?
- How do I correct an error?

Tomorrow

eXplainable AI

Data → eXplainable AI Model → eXplainable AI Product → Decision / Explanation

Feedback

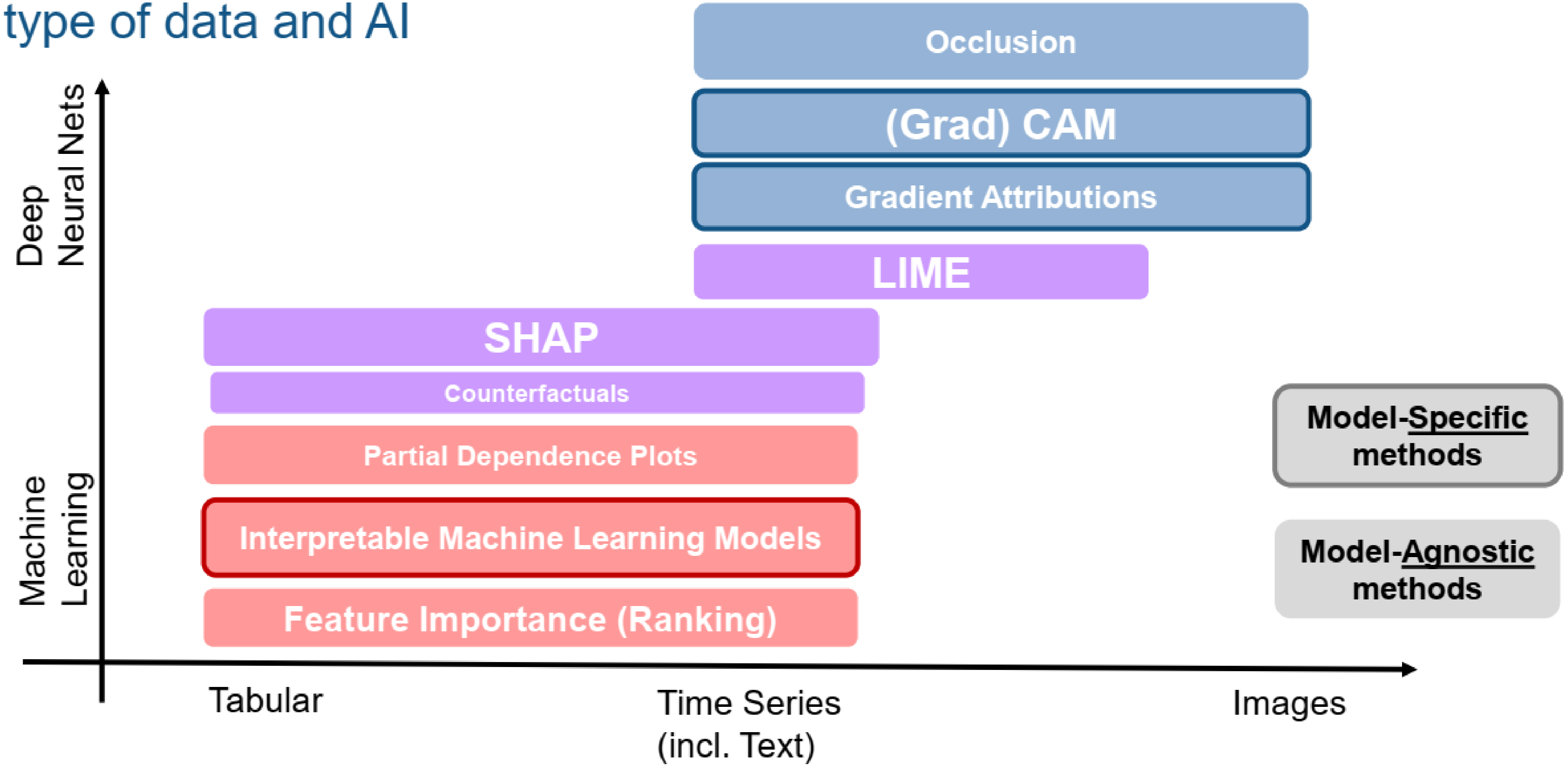
- I understand why
- I understand why not
- I know why you succeed or fail
- I understand, so I trust you

[MathWorks Online Seminar: eXplainable AI and AI V&V]

Unexplainable vs. explainable AI

- Techniques and methods for human-understandable explanations for the behaviour of AI systems
    - Feature importance
        - By identifying the most relevant features or inputs that contributed to a model's prediction
    - Model interpretability
        - By developing inherently interpretable models
    - Visual explanations
        - By highlighting the regions or features in the input data that were most influential in a model's decision-making process using visualizations

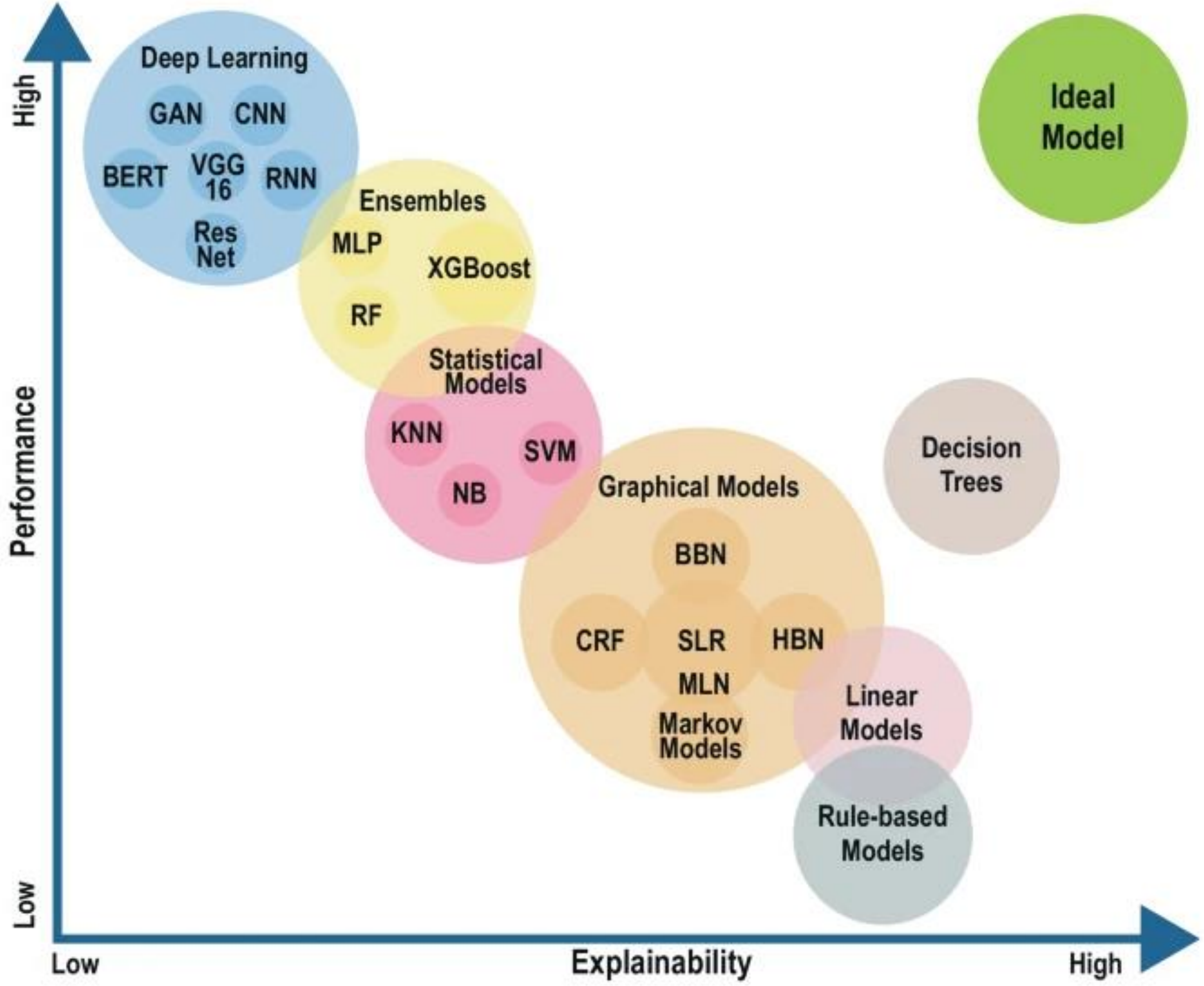
Wide range of techniques with varying applicability depending on type of data and AI



[MathWorks Online Seminar: eXplainable AI and AI V&V]

Techniques for XAI

- Challenges
  - Trade-offs between model complexity, performance, and interpretability
  - Appropriate forms and levels of explanations required for different applications and user groups



[Viswan et al., 2024]

Trade-off between model accuracy and associated explainability

SHAP (SHapley Additive exPlanations)

- Game-theoretic approach to explain the output of machine learning models
  - Particularly grounded in the Shapley value that distributes the total gain generated by the coalition of all players (features) fairly among them
- SHAP values provide a unified measure of feature importance by distributing the prediction among the features, indicating how much each feature contributes to a model's output

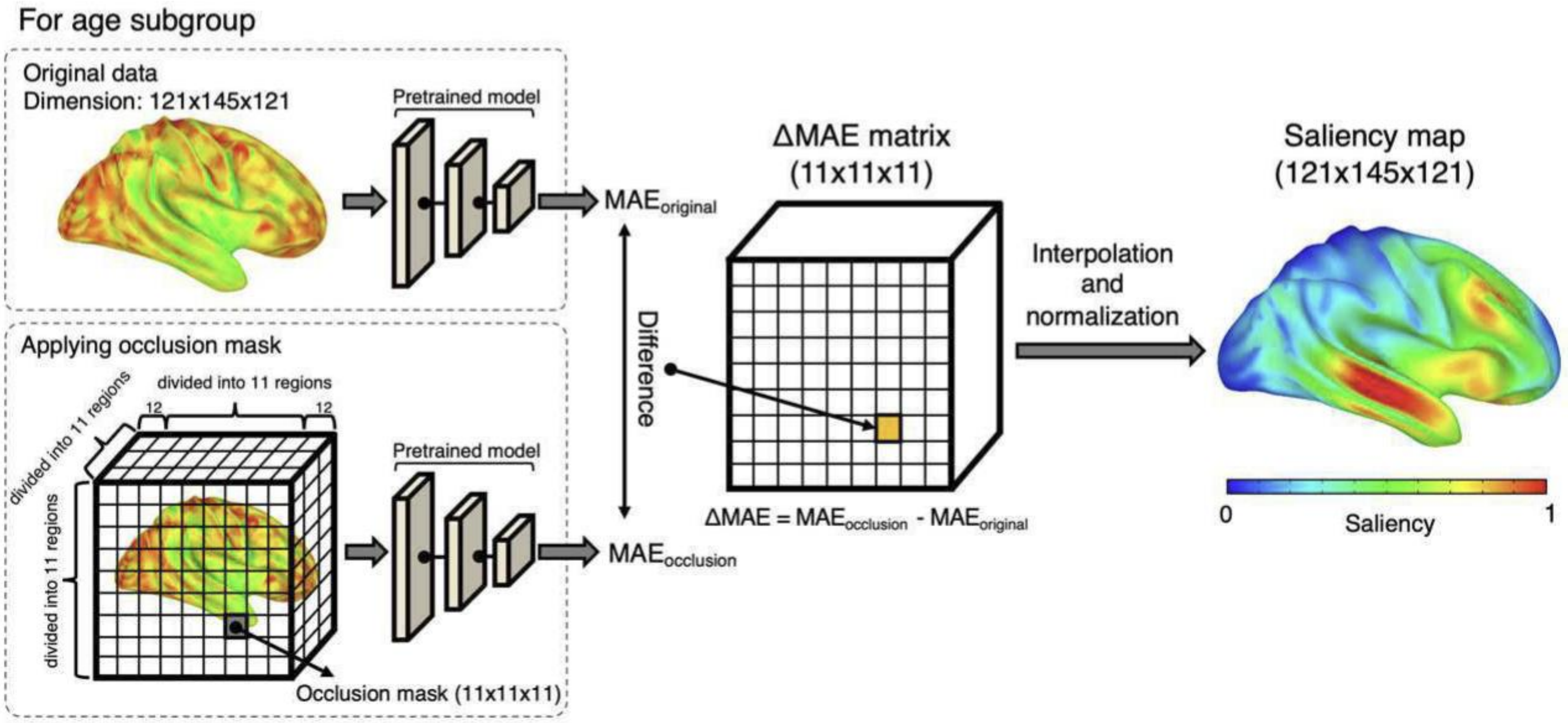
- Key features
  - Interpretability
    - SHAP values offer a consistent and interpretable method to understand how each feature influences a model’s prediction
  - Additivity
    - SHAP values ensure that the contributions of all features add up to the difference between the actual prediction and the average prediction, making them additive
  - Fairness
    - SHAP values provide a fair allocation of the contribution of each feature to the prediction by considering all possible feature combinations

  - Consistency
    - If a model changes so that a feature contributes more, its SHAP value also increases, ensuring consistent feature importance
- Types of SHAP explainers
  - Kernel SHAP
    - Model-agnostic approach that can explain any machine learning model
  - Tree SHAP
    - Optimized for tree-based models, providing fast and accurate SHAP values
  - Deep SHAP
    - Designed for deep learning models
  - Linear SHAP
    - For linear models

Occlusion Sensitivity Analysis

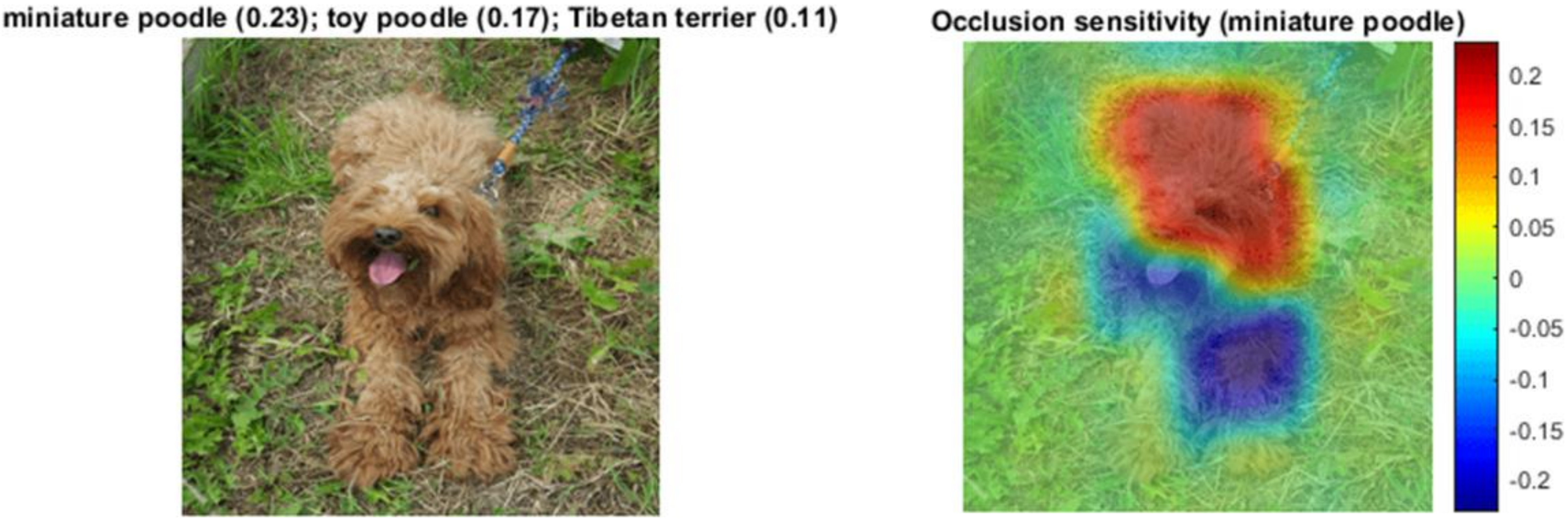
- Technique for understanding the importance of different regions in an image for a model's prediction
- Based on the idea that image regions that cause a larger change in a model's prediction when occluded are considered more important or sensitive for the task at hand
- Computationally expensive, especially for large images and models

- Occlusion sensitivity map (saliency map)
  - Computed by perturbing small regions of an image by replacing it with an occluding mask, typically a zero-valued mask, moving the mask across the image, and recording a change in a model's prediction for each occluded region
  - Represents a model's sensitivity to occlusion in different regions of an image
  - Highlights which parts of an image are most important to a model's prediction



[Lee et al., 2022]

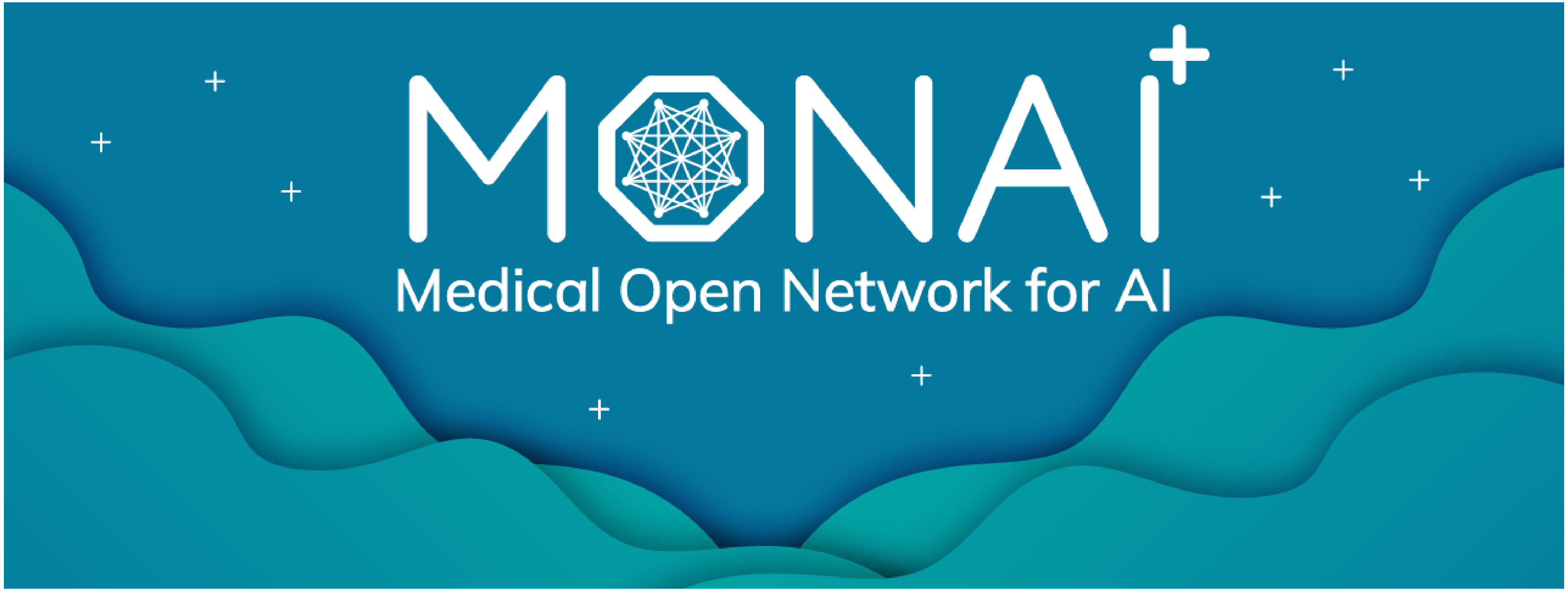
Occlusion sensitivity analysis for explaining image regions' contributions



[https://www.mathworks.com/help/deeplearning/ug/understand-network-predictions-using-occlusion.html]

Occlusion sensitivity map showing image regions' contributions

Medical Open Network for AI (MONAI)



[https://github.com/Project-MONAI]

- Project MONAI
  - Initiative started initially by NVIDIA and King's College London to establish an inclusive community of AI researchers to develop and exchange best practices for AI in healthcare imaging across academia and enterprise researchers
    - Evolved into a growing consortium currently with 16 different universities and industrial partners
  - Aimed to define, standardize, develop, and exchange best practices for AI in healthcare by unifying the fragmented healthcare AI software field resulted from the disjointed development of several platforms such as NiftyNet [Gibson, et al., 2018], DLTK [Pawlowski, et al., 2017], DeepNeuro [Beers, et al., 2021], NVIDIA Clara [https://www.nvidia.com/en-us/clara/], and Microsoft Project InnerEye [https://www.microsoft.com/en-us/research/project/medical-image-analysis/]

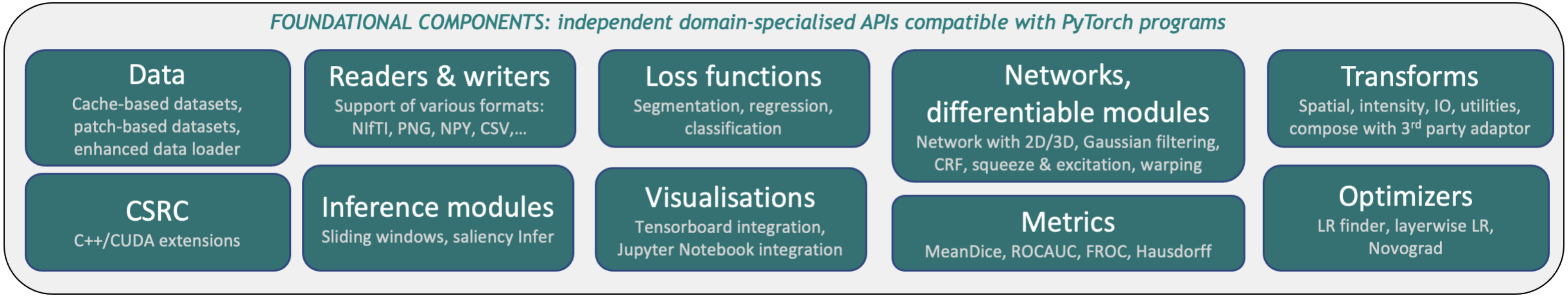
- Released multiple open-source PyTorch-based frameworks for annotating, building, training, deploying, and optimizing AI workflows in healthcare
  - MONAI Core: for training AI models for healthcare imaging
  - MONAI Label: for quickly annotating new datasets
  - MONAI Deploy App ADK: for integrating AI models into clinical workflows
  - MONAI Model Zoo: for sharing a collection of medical imaging models in the MONAI Bundle format

- MONAI Core
  - PyTorch-based, open-source, freely available, community-supported, and consortium-led framework (suite of libraries, tools, and SDKs) for deep learning in healthcare
    - Extends PyTorch to support medical data, with a particular focus on imaging, video, and other forms of structured data, as part of PyTorch ecosystem (tools, libraries, and more built up around the core PyTorch functionality to support, accelerate, and explore AI development)

- Provides field-specific and domain-optimized functionality that is not often supported by general-purpose frameworks such as Tensorflow and PyTorch
  - Medical images are often stored in complex formats with rich meta-information, with the data volumes being high-dimensional and requiring carefully designed manipulation procedures
  - If healthcare AI models are to be built using a general-purpose framework, significant functionality would need to be developed and tested, thus increasing the length of and the risks associated with the full research and development life-cycle
- Provides wrappers and adaptors that allow popular healthcare AI tools to be used from within MONAI
  - Developed with minimal required dependencies, namely PyTorch and NumPy

– Modules

- **`monai.data`**: datasets, readers/writers, and synthetic data
- **`monai.losses`**: classes defining loss functions
- **`monai.networks`**: network definitions, component definitions, and PyTorch-specific utilities
- **`monai.transforms`**: classes defining data transforms for pre-processing and post-processing
- **`monai.csrc`** and **`monai.extensions`**: C++/CUDA extensions for MONAI core Python APIs
- **`monai.visualize`**: utilities for data visualization
- **`monai.metrics`**: metric tracking and analysis tools
- **`monai.optimizers`**: classes defining optimizers



FOUNDATIONAL COMPONENTS: independent domain-specialised APIs compatible with PyTorch programs

Data
Cache-based datasets, patch-based datasets, enhanced data loader

Readers & writers
Support of various formats: NIfTI, PNG, NPY, CSV,...

Loss functions
Segmentation, regression, classification

Networks, differentiable modules
Network with 2D/3D, Gaussian filtering, CRF, squeeze & excitation, warping

Transforms
Spatial, intensity, IO, utilities, compose with 3rd party adaptor

CSRC
C++/CUDA extensions

Inference modules
Sliding windows, saliency Infer

Visualisations
Tensorboard integration, Jupyter Notebook integration

Metrics
MeanDice, ROCAUC, FROC, Hausdorff

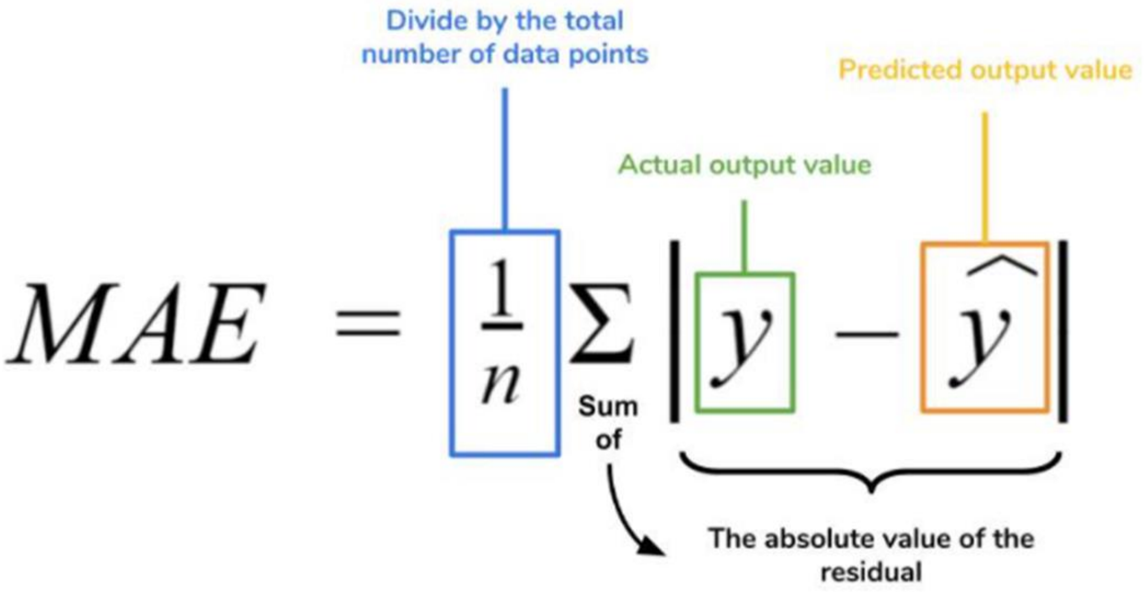
Optimizers
LR finder, layerwise LR, Novograd

[Cardoso et al., 2021]

MONAI Core modules

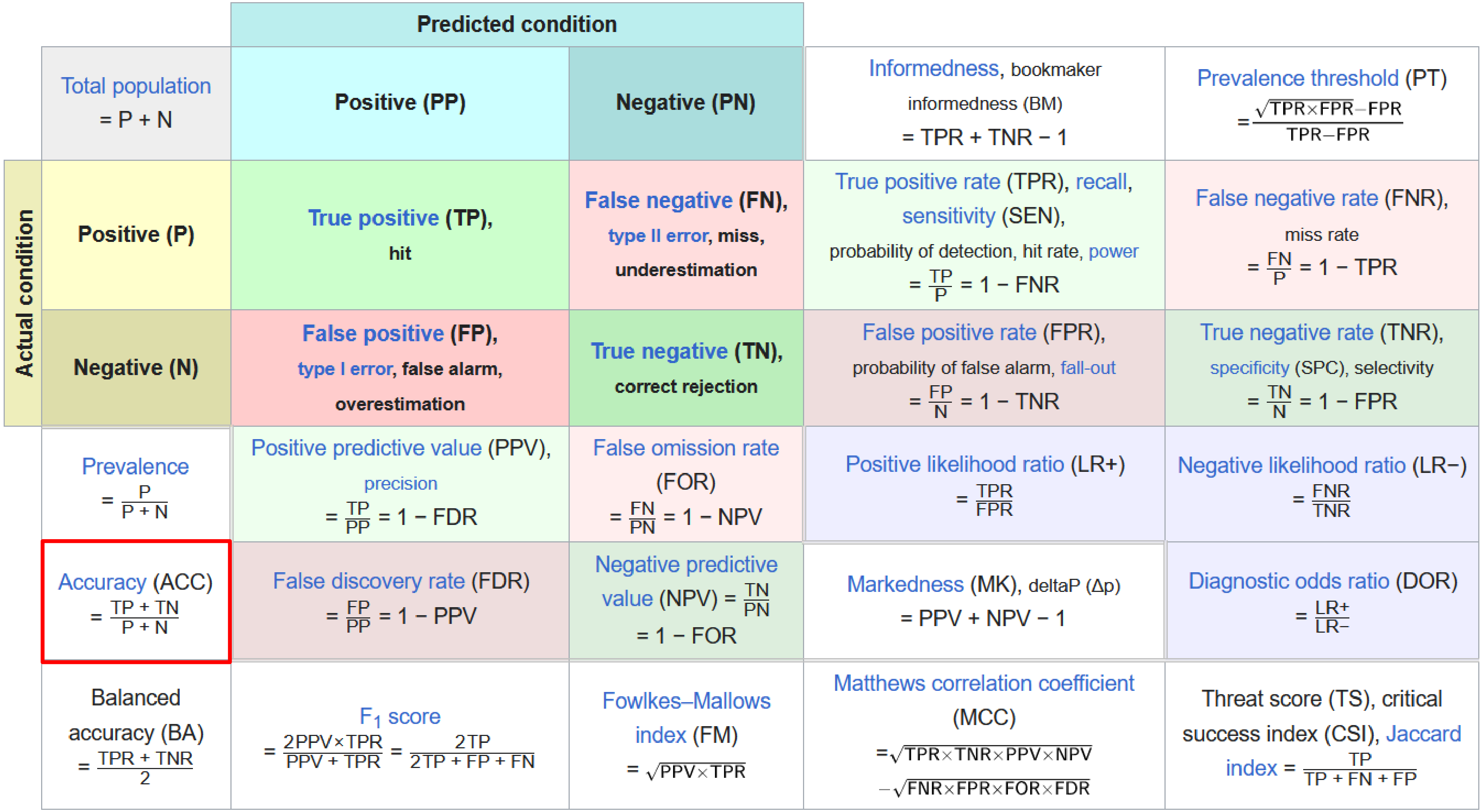
Model Performance Metrics

- Predicting memory performance
  - Mean absolute error (MAE)
    - (sum of absolute errors)/(sample size)



Divide by the total number of data points
Predicted output value
Actual output value
$MAE = \frac{1}{n}\sum\left|y - \hat{y}\right|$
Sum of
The absolute value of the residual

- Predicting sex
  - Accuracy
    - Proportion of correct predictions



		Predicted condition			
	Total population $= P + N$	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP),** **hit**	**False negative (FN),** **type II error, miss,** **underestimation**	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP),** **type I error, false alarm,** **overestimation**	**True negative (TN),** **correct rejection**	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN}$ $= 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

[https://en.wikipedia.org/wiki/Confusion_matrix]

Accuracy in a confusion matrix

Score Evaluation (5 Points)

- Model performance on the test set
  - In predicting memory performance, MAE for 10 individuals in the test set
  - In predicting sex, accuracy for 50 individuals in the test set

- Effective utilization of multimodal data
  - Grey matter and white matter maps/features from structural MRI
  - Regional homogeneity and default mode network maps/features from resting state functional MRI
  - Fractional anisotropy and mean diffusivity maps/features from diffusion-weighted MRI
- Model explanations
  - To better understand the reasoning behind a model's prediction: how much each brain region contributes to a model's prediction
  - To identify and mitigate potential biases or errors