RAI Documentation

Release 0.1

RAI Contributers

Oct 07, 2023

USER GUIDE:

RAI is a python library that is written to help AI developers in various aspects of responsible AI development. It consists of a core API and a corresponding web-based dashboard application. RAI can easily be integrated into AI development projects and measures various metrics for an AI project during each phase of AI development, from data quality assessment to model selection based on performance, fairness and robustness criteria. In addition, it provides interactive tools and visualizations to understand and explain AI models and provides a generic framework to perform various types of analysis including adversarial robustness.

Note:

The dashboard GUI is currently designed around 1920x1080 displays

CHAPTER

ONE

INTRODUCTION

- RAI is an easy-to-use framework for Responsible AI development. By providing models and their generations, RAI can handle all metric computation and perform live visualization on its dashboard.
- RAI can also be used entirely from the console, allowing users working through command line to take advantage of its tools.
- RAI is built to handle a large variety of models and datatypes like text, images, tabular data, etc.
- Based on the type of model, data, and task provided by the user, RAI automatically determines what metrics are relevant.
- Visualization tools built into RAI give users a strong sense of how their model performs and allows for in depth analysis and optimization.
- RAI provides metrics that measure various aspects of AI systems, including performance, robustness, explainability, and fairness.
- Emphasis was placed on making each metric simple to understand and visualize, allowing anyone to get a strong idea of their model's strengths and weaknesses and understand what needs to change.

CHAPTER

TWO

GETTING STARTED

Here's a quick example of using RAI without a dashboard for calculating and reporting on machine learning metrics.

- It starts by importing the necessary libraries

Fig. 1: Getting_started_demo

CHAPTER

THREE

INSTALLATION

3.1 Windows 10

We recommend using Visual studio code for Windows users for easier installation of Python packages and required libraries. For this RAI project you need an environment with Python version 3.9.

Some packages uses Visual C++ 14.0 BuildTools. You can also install the build tools from Microsoft Visual studio code . The build tools do not come with Visual Studio Code by default.

- **Setting up the Documentation sources**.

VS code

https://code.visualstudio.com/download

Python Version

3.9.13- https://www.python.org/downloads/windows/

Pip Version

22.3

Clone Git-Repo

https://github.com/cisco-open/ResponsibleAI.git

Note: `NumPy 1.23.4` latest version is not compatible with python 3.10 version.

3.2 Install a package locally and run.

Here is a quick demo of how to install a package locally and run in the environment:

- Install packages in requirement.txt file.

```
Run pip install ``-r requirements.txt``.
```

Warning: If you run into any Error. For instance:Package could not install `plotly`. Install `plotly` separately with the following command `python pip install 'plotly' --user`.

- Try installing packages in requirements.txt again
- pip install -r requirements.txt.

All packages are successfully installed.

- RAI can then be installed using.

```
pip install py-rai
```

Description: when you are developing it on your system any changes to the original package would reflect directly in your environment.

CHAPTER

FOUR

RESPONSIBLE AI DEVELOPMENT IN RAI

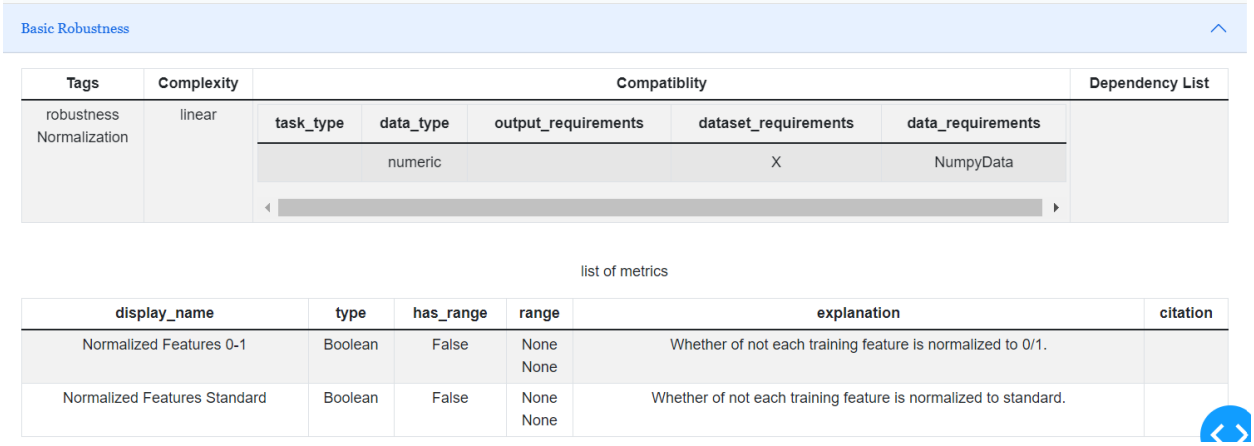
As the use of AI in various segments of the industry rapidly grows, responsible AI is garnering increased attention from both the AI research community and the industry. In fact, many AI pioneers and well-known advocates have emphasized the need for establishing a reliable, fair, transparent, and robust set of standards and practices for AI development. Responsible AI (RAI) has been designed to simplify the evaluation of AI models for developers and data scientists from various perspectives of responsible AI development. RAI offers a unified framework for evaluating AI models not only based on performance but also considering bias and fairness, robustness, and trustworthiness of AI models. To do this, it gathers various types of metrics from multiple open-source libraries in an easy-to-use and interactive manner. Below, we outline some of the metric categories related to responsible AI development.

Basic Robustness

- Model robustness refers to the degree that a model's performance changes when using new data versus training data.

display_name	Description
Normalized Features 0-1	Whether of not each training feature is normalized to 0/1.
Normalized Features Standard	Whether of not each training feature is normalized to standard.

Basic Robustness

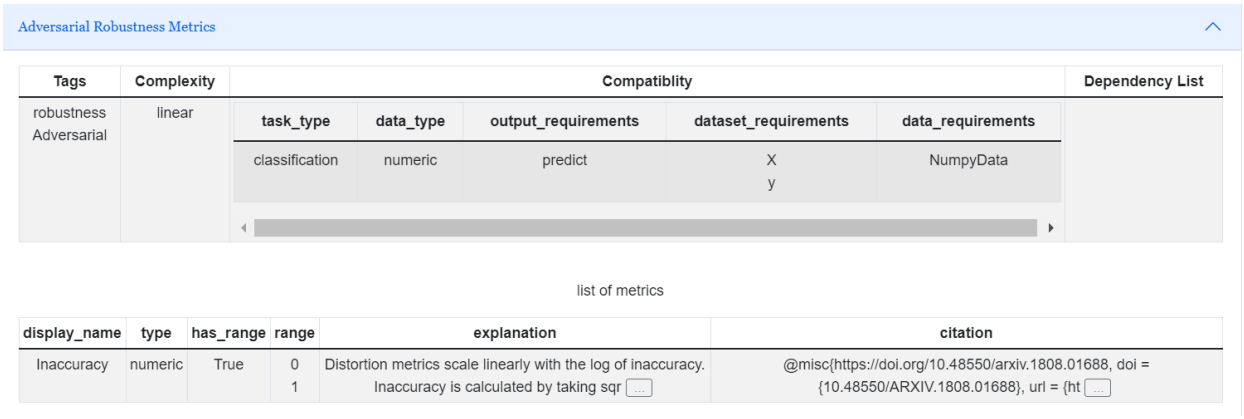


Adversarial Robustness

- Adversarial robustness is the ability of an AI model to resist being fooled by a series of carefully crafted changes to its input data.

display_name	Description
Accuracy	Distortion metrics scale linearly with the log of inaccuracy. Inaccuracy is calculated by taking sqrt(1 - accuracy).

Adversarial Robustness

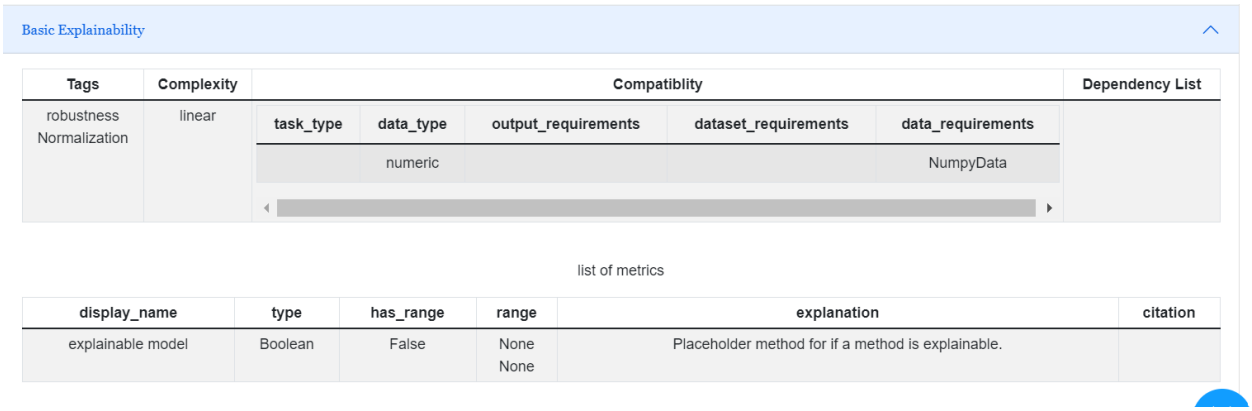


4.1 Basic Explainability

- Basic Explainability will assist in providing details on or causes of fairness metrics.

display_name	Description
explainable model	Placeholder method for if a method is explainable.

basic Explainability



4.2 Performance

- Performance metrics are a part of every machine learning pipeline. They tell you if you're making progress, and put a number on it. All machine learning models, whether it's linear regression, or a SOTA technique like BERT, need a metric to judge performance.
- The Torch library is used in our implementation of performance metrics in order to take advantage of some of its features.

Performance Metrics

display_name	Description
Accuracy	The proportion of correct predictions among the total number of cases processed.
Balanced Accuracy	Describes the proportion of correct predictions averaged across each label.
False Positive Rate	Describes the percentage of negative examples incorrectly predicted to be positive.

Classification Performance Metrics

Classification Performance Metrics

Tags	Complexity	Compatiblity					Dependency List
performance Classification	linear	task_type	data_type	output_requirements	dataset_requirements	data_requirements	
		classification		predict	y		

list of metrics

display_name	type	tags	has_range	range	explanation	citation
Accuracy	numeric		True	0 1	The proportion of correct predictions among the total number of cases processed. The ideal value is ...	@article{scikit-learn, title={Scikit-learn: Machine Learning in {P}ython}, author={Pedregosa ...
Balanced Accuracy	numeric		True	0 1	Describes the proportion of correct predictions averaged across each label. The ideal value is 1.0.	@article{scikit-learn, title={Scikit-learn: Machine Learning in {P}ython}, author={Pedregosa ...
Confusion Matrix	Matrix		False	None	A Confusion Matrix C is a matrix C such that C(i,j) is equal to the number of observations known to ...	@article{scikit-learn, title={Scikit-learn: Machine Learning in {P}ython}, author={Pedregosa ...
F1 Score	Vector_no_display		True	0 1	Can be interpreted as a harmonic mean of the precision and recall. The formula for the F1 score is: ...	@article{scikit-learn, title={Scikit-learn: Machine Learning in {P}ython}, author={Pedregosa ...
Average F1 Score	numeric		True	0	Can be interpreted as a harmonic mean of the precision	@article{scikit-learn, title={Scikit-learn: Machine

4.3 Fairness

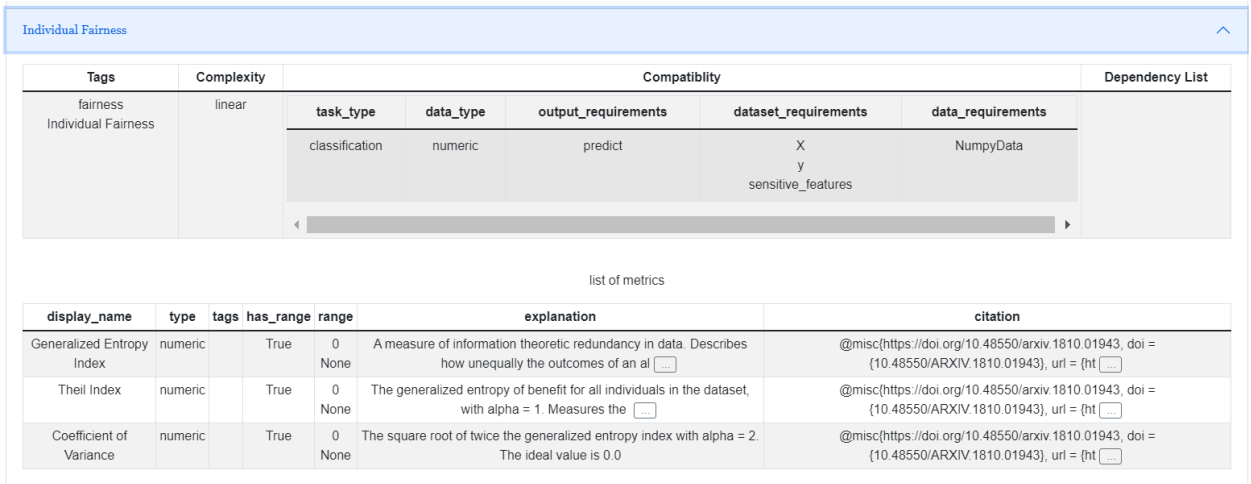
- Fairness measures allow us to assess and audit for possible biases in a trained model. There are several types of metrics that are used in RAI to assess a model's fairness. They can be classified as follows:

Individual Fairness

- The goal of similar individuals receiving similar treatments or outcomes.It is used to compute metrics related to individual fairness in AI system.

display_name	Description
generalized_entropy_in	A measure of information theoretic redundancy in data. Describes how unequally the outcomes of an algorithm benefit different individuals or groups in a population
theil_index	The generalized entropy of benefit for all individuals in the dataset, with alpha = 1.nMeasures the inequality in benefit allocation for individuals. nA value of 0 implies perfect fairness
coefficient_of_variati	The square root of twice the generalized entropy index with alpha = 2. nThe ideal value is 0.

Individual Fairness

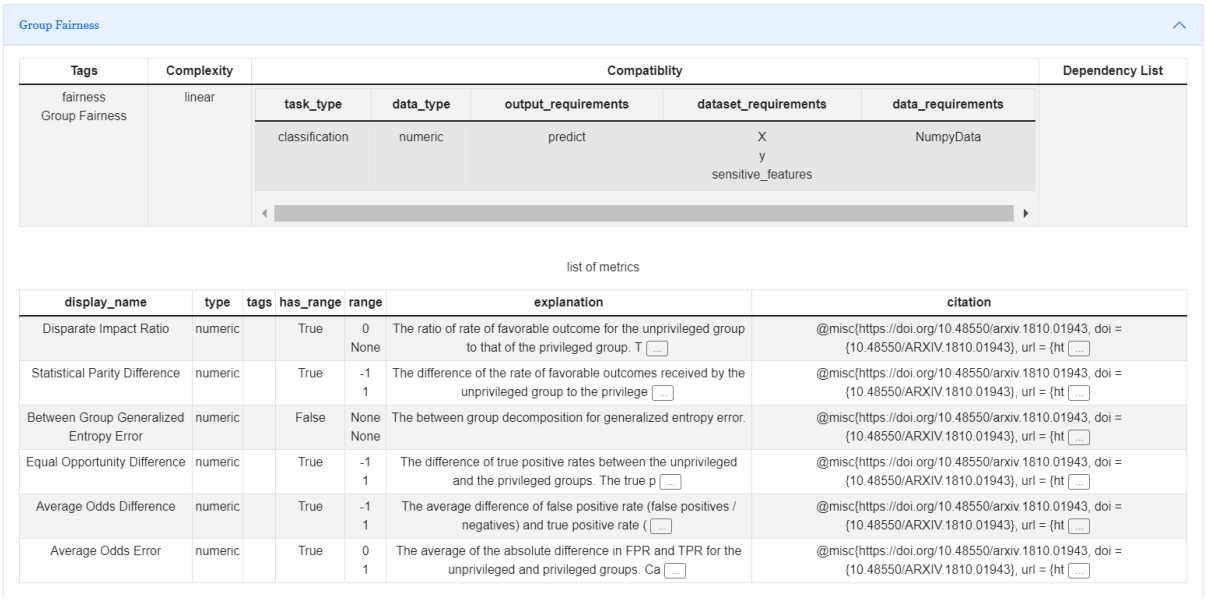


Group Fairness

- It is the goal of groups defined by protected attributes to receive similar treatment or outcomes regardless of their protected attributes.

display_name	Description
disparate_impact	The ratio of rate of favorable outcome for the unprivileged group to that of the privileged group. nThe ideal value of this metric is 1.0 A value < 1 implies higher benefit for the privileged group and a value > 1 implies a higher benefit for the unprivileged group.
statistical_parity_dif	The difference of the rate of favorable outcomes received by the unprivileged group to the privileged group. nThe idea value is 0.0
between_group_	The between group decomposition for generalized entropy error
equal_opportu	The difference of true positive rates between the unprivileged and the privileged groups. nThe true positive rate is the ratio of true positives to the total number of actual positives for a given group. nThe ideal value is 0. A value of < 0 implies higher benefit for the privileged group and a value > 0 implies higher benefit for the unprivileged group

Group fairness



Group Fairness

Tags	Complexity	Compatiblity					Dependency List
		task_type	data_type	output_requirements	dataset_requirements	data_requirements	
fairness Group Fairness	linear	classification	numeric	predict	X y sensitive_features	NumpyData	

list of metrics

display_name	type	tags	has_range	range	explanation	citation
Disparate Impact Ratio	numeric		True	0 None	The ratio of rate of favorable outcome for the unprivileged group to that of the privileged group. T ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Statistical Parity Difference	numeric		True	-1 1	The difference of the rate of favorable outcomes received by the unprivileged group to the privilege ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Between Group Generalized Entropy Error	numeric		False	None None	The between group decomposition for generalized entropy error.	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Equal Opportunity Difference	numeric		True	-1 1	The difference of true positive rates between the unprivileged and the privileged groups. The true p ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Average Odds Difference	numeric		True	-1 1	The average difference of false positive rate (false positives / negatives) and true positive rate (...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Average Odds Error	numeric		True	0 1	The average of the absolute difference in FPR and TPR for the unprivileged and privileged groups. Ca ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...

General Prediction Fairness

- For the classification model to be fair, various fairness metrics need to be computed..

display_name	Description
average_odds_differ	The average difference of false positive rate (false positives / negatives) and true positive rate (true positives / positives) between unprivileged and privileged groups. nThe ideal value is 0. A value of < 0 implies higher benefit for the privileged group and a value > 0 implies higher benefit for the unprivileged group
between_all_group	The square root of twice the pairwise entropy between every pair of privileged and underprivileged groups with alpha = 2.nThe ideal value is 0
between_all_group	The pairwise entropy between every pair of privileged and underprivileged groups. nThe ideal value is 0.0
between_all_group	The pairwise entropy between every pair of privileged and underprivileged groups with alpha = 1.nThe ideal value is 0.0

General prediction Fairness

Tags	Complexity	Compatiblity					Dependency List
		task_type	data_type	output_requirements	dataset_requirements	data_requirements	
fairness General Fairness	linear	classification	numeric	predict	X y sensitive_features	NumpyData	

list of metrics

display_name	type	tags	has_range	range	explanation	citation
Average Odds Difference	numeric		True	0 1	The average difference of false positive rate (false positives / negatives) and true positive rate (...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Between all Groups Coefficient of Variation	numeric		True	0 None	The square root of twice the pairwise entropy between every pair of privileged and underprivileged g ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Between all Groups Generalized Entropy Index	numeric		True	0 None	The pairwise entropy between every pair of privileged and underprivileged groups. The ideal value is ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Between all Groups Theil Index	numeric		True	0 None	The pairwise entropy between every pair of privileged and underprivileged groups with alpha = 1. The ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Between Group Coefficient of Variation	numeric		False	None None	The square root of twice the pairwise entropy between a given pair of privileged and underprivileged ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Between Group Generalized Entropy Index	numeric		True	0 None	The pairwise entropy between a given pair of privileged and underprivileged groups. The ideal value ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Between Group Theil Index	numeric		True	0 None	The pairwise entropy between a given pair of privileged and underprivileged groups with alpha = 1. T ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Coefficient of Variation	numeric		True	0 None	The square root of twice the generalized entropy index with alpha = 2. The ideal value is 0.0.	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Consistency	numeric		False	None None	A measure how similar the labels are for similar instances. The ideal value is 1.0	@inProceedings{pmlr-v28-zemel13, title = {Learning Fair Representations}, author = {Zemel, ...
Differential Fairness Bias Amplification	numeric		False	None None	The difference in smoothed EDF between the classifier and the original dataset. Positive values mea ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Error Rate	numeric		True	0 1	The percentage of predictions that were incorrect. Computed as (1 -(true positive count + true negat ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Error Rate Difference	numeric		True	-1 1	The difference of error rates between unprivileged and privileged groups. Where Error Rate is the pe ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Error Rate Ratio	numeric		True	0 None	The ratio of error rates between unprivileged and privileged groups. Where Error Rate is percentage ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
False Discovery Rate	numeric		True	0 1	The percentage of positive predictions that were false positives. Calculated as (false positive coun ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...

Dataset Fairness

- It is used to compute fairness metrics for the Binary dataset.

display_name	Description
base_rate	Base Rate is the rate at which a positive outcome occurs in Data. In formula it is, Pr(Y=pos_label) = P/(P+N)
num_instances	Num Instances counts the number of examples in Data
num_negatives	Num Negatives counts the number of negative labels in Data
num_positives	Num Positives calculates the number of positive labels in Data

Dataset Fairness

Dataset Fairness

Tags	Complexity	Compatiblity					Dependency List
		task_type	data_type	output_requirements	dataset_requirements	data_requirements	
fairness Data Fairness	linear	classification	numeric		X sensitive_features	NumpyData	

list of metrics

display_name	type	tags	has_range	range	explanation	citation
Base Rate	numeric		True	0 1	Base Rate is the rate at which a positive outcome occurs in Data. In formula it is, Pr(Y=pos_label) ...	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Num Instances	numeric		True	0 None	Num Instances counts the number of examples in Data.	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Num Negatives	numeric		True	0 None	Num Negatives counts the number of negative labels in Data.	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...
Num Positives	numeric		True	0 None	Num Positives calculates the number of positive labels in Data.	@misc{https://doi.org/10.48550/arxiv.1810.01943, doi = {10.48550/ARXIV.1810.01943}, url = {ht ...

For Instance:

- Using RAI to measure group fairness:

Fig. 1: fairness_of_the_model

A case study of how RAI can be used to detect and resolve biases in AI models can be found *here*.

CHAPTER

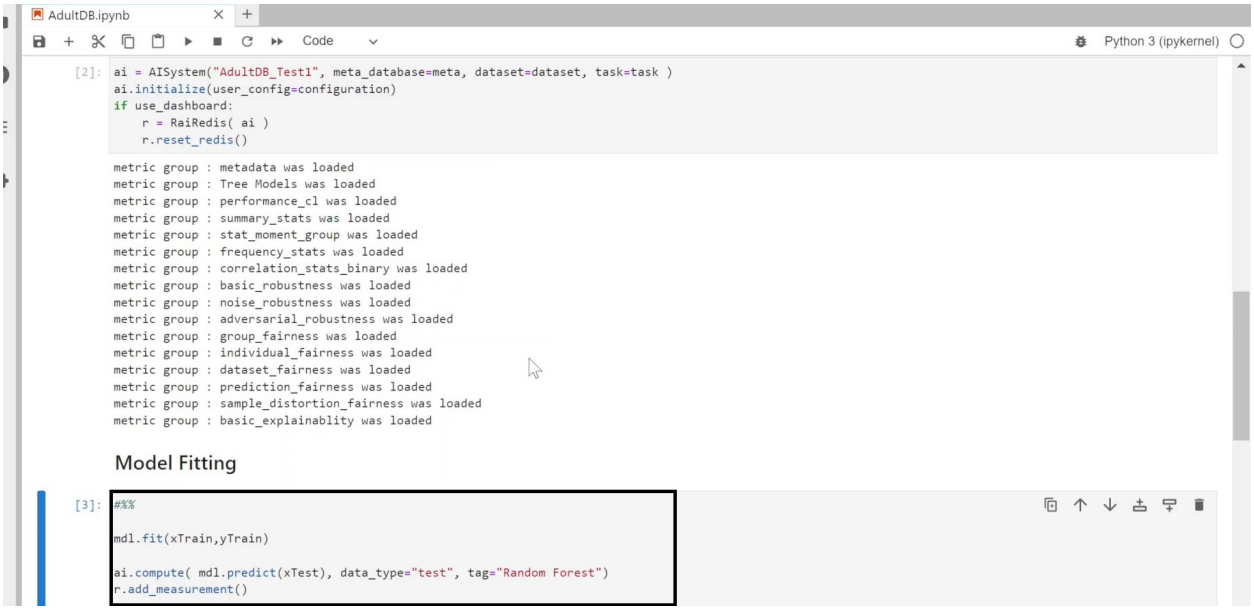
FIVE

ROBUSTNESS OF AI

In this Demo case, we can see how RAI can detect and resolve bias and fairness in AI models.

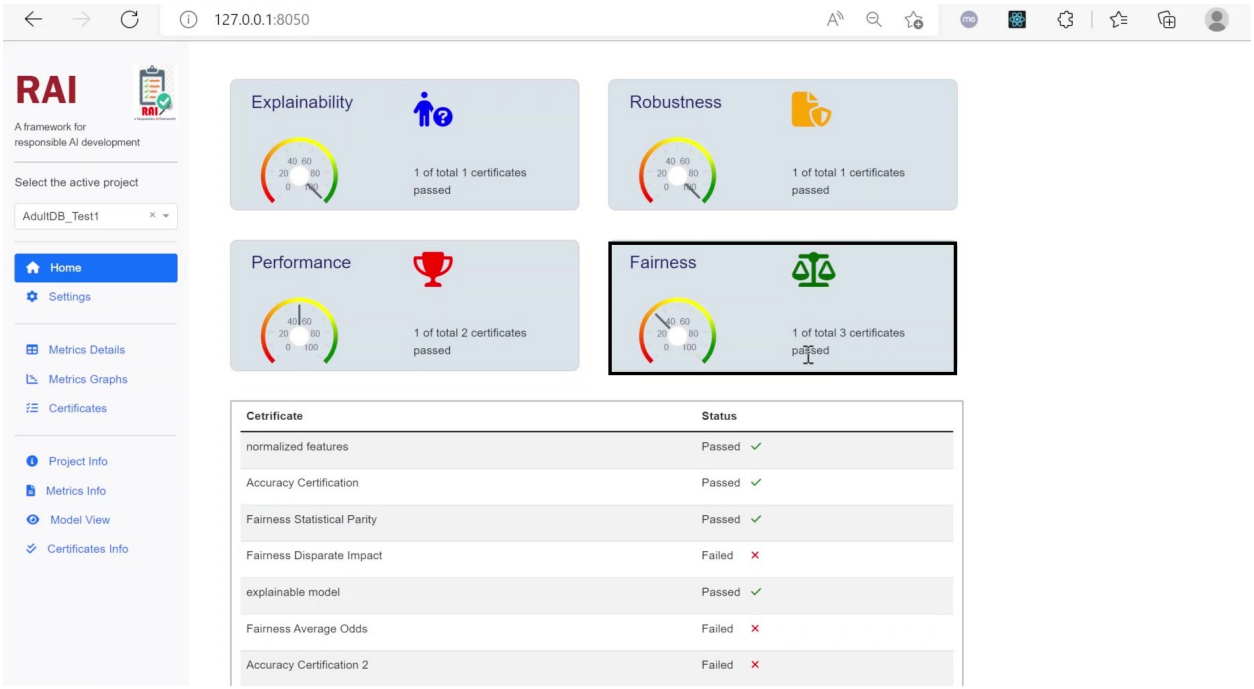
- To demonstrate how RAI works, let's consider a simple data science project to predict the income level of participants.
- In this dataset, there is an imbalance between white and black participants.
- Here RAI will show how to identify and mitigate the problem.
- After fitting the model, we can ask RAI to send the measurements back to the dashboard.

fitting the model



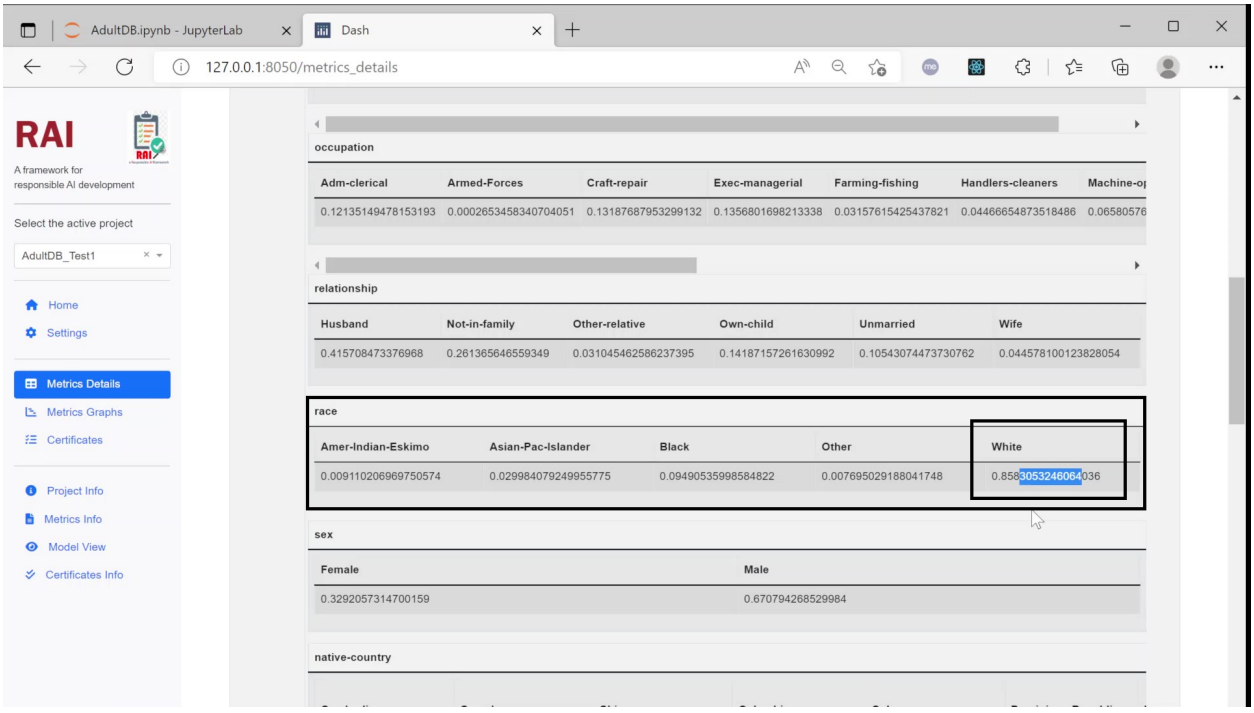
- We can now go back to the dashboard and see how the system has performed for each category.
- For instance, we can see that 1 out of 3 tests is passed for fairness. This shows a significant problem in fairness.

significant problem in fairness



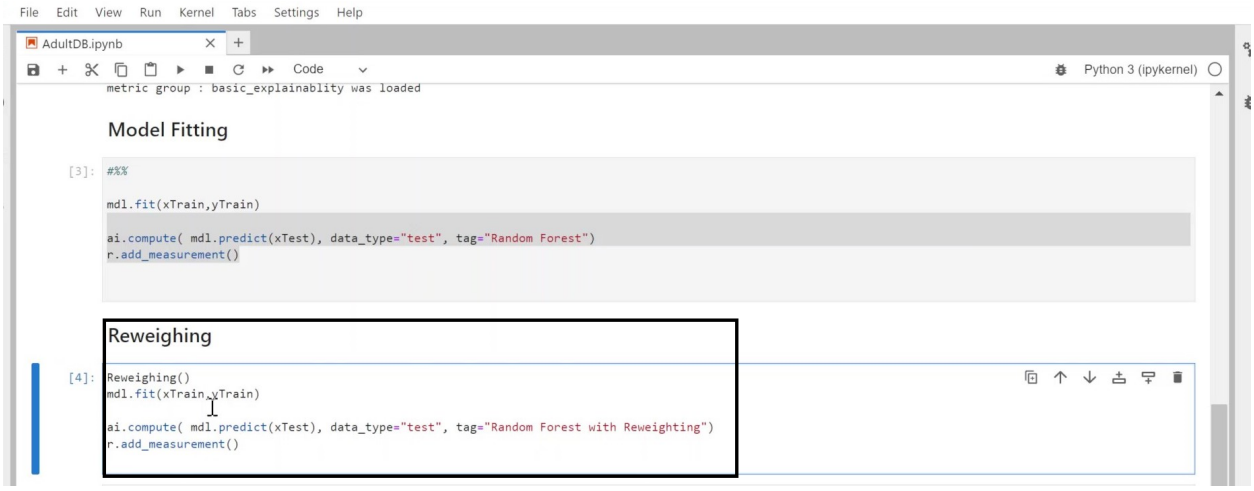
- Now we can investigate this problem by looking at the individual metrics.
- We can select the category of interest, and for each category, we can look at the individual metric that has been calculated.
- For instance, we can go to frequency statistics and look at the race parameter, which shows more than 85% of participants are white.

race parameter



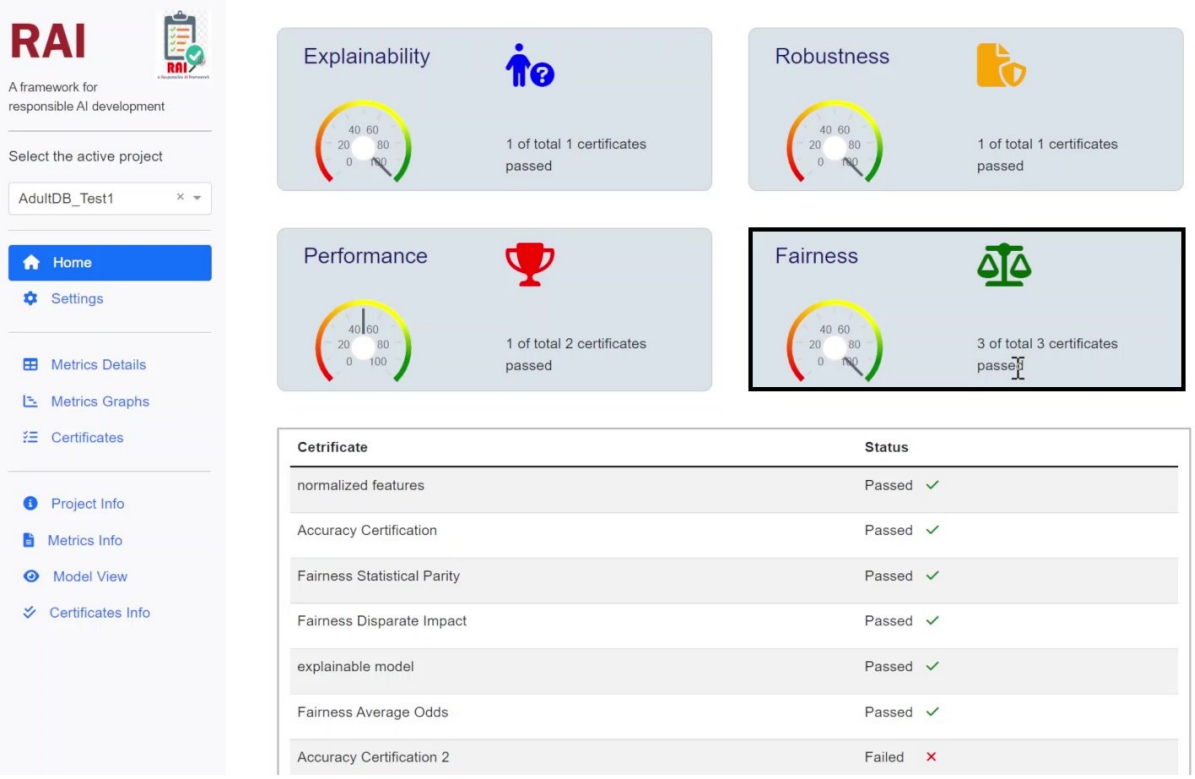
- To mitigate this imbalance problem, we can go back to the data science project and perform some mitigation strategies.
- Here we are using Reweighing algorithm after fitting the model once again.
- We can ask RAI to compute the metrics again and evaluate our model.

Reweighing algorithm



- Now we can go back to the dashboard.
- At the dashboard's homepage, we can look at how the system has performed after this mitigation, which shows that all the fairness tests have passed this time.

fairness tests have passed



CHAPTER

SIX

MODEL SELECTION

Model selection is the process of selecting one of the models as the final ML model for a training dataset.

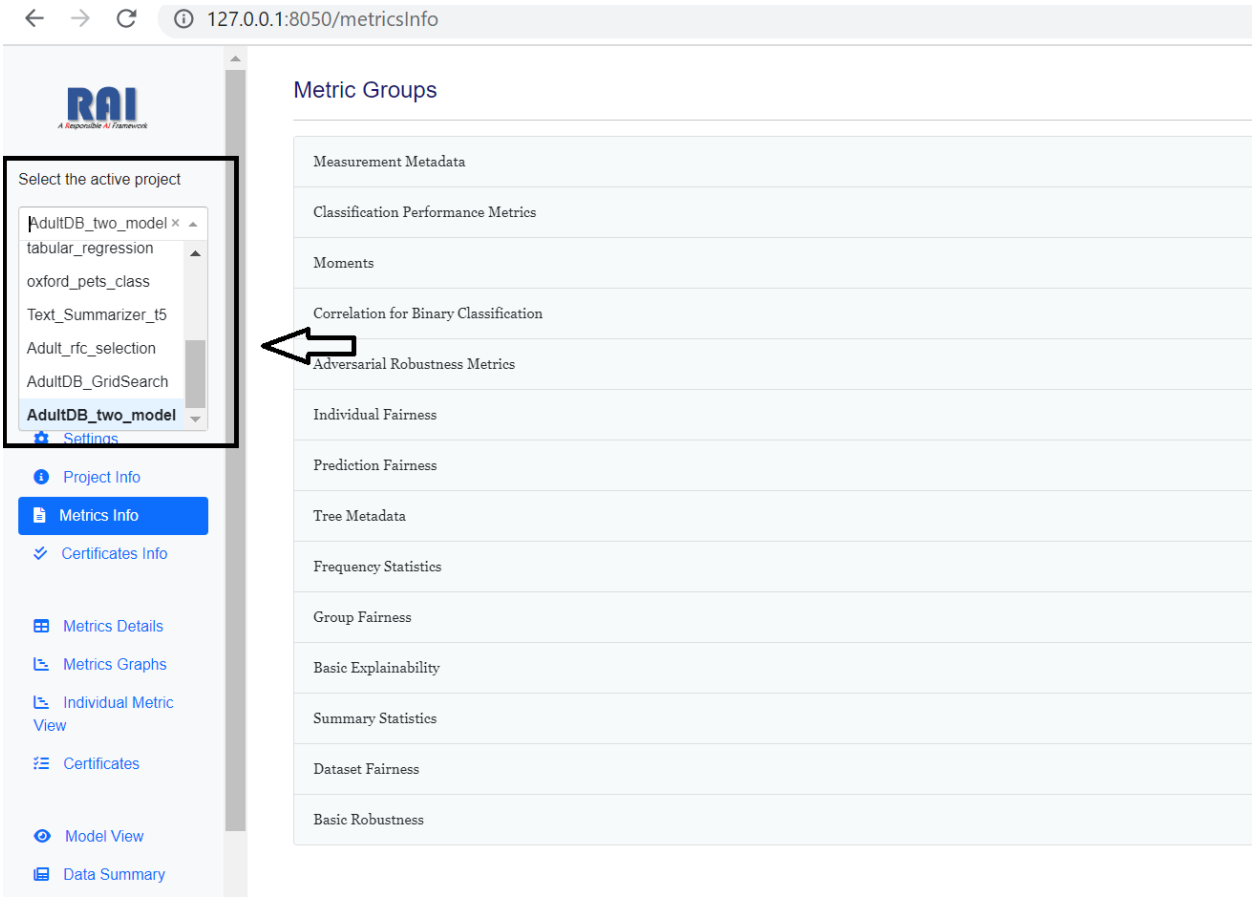
- To figure this out, RAI will usually come up with some kind of evaluation metric.
- Then it will divide the training dataset into three parts: A training set, a Validation set(sometimes called development), and a Test dataset.
- **The Training** - It is used to fit the models,
- **The Validation** - It is used to estimate prediction error for model selection,
- **The Test set** - It is used to do a final evaluation and assessment of the generalization error of the chosen model on the test dataset.
- This way, we can determine the model with the lowest generalization error. It refers to the performance of the model on unseen data, i.e., data that the model hasn't been trained on.

Example

We may have a dataset for which we are interested in visualizing the performance of the individual case. We do not know beforehand as to which model will perform best on this problem, as it is unknowable. Therefore, we fit and evaluate a suite of different models for the problem.

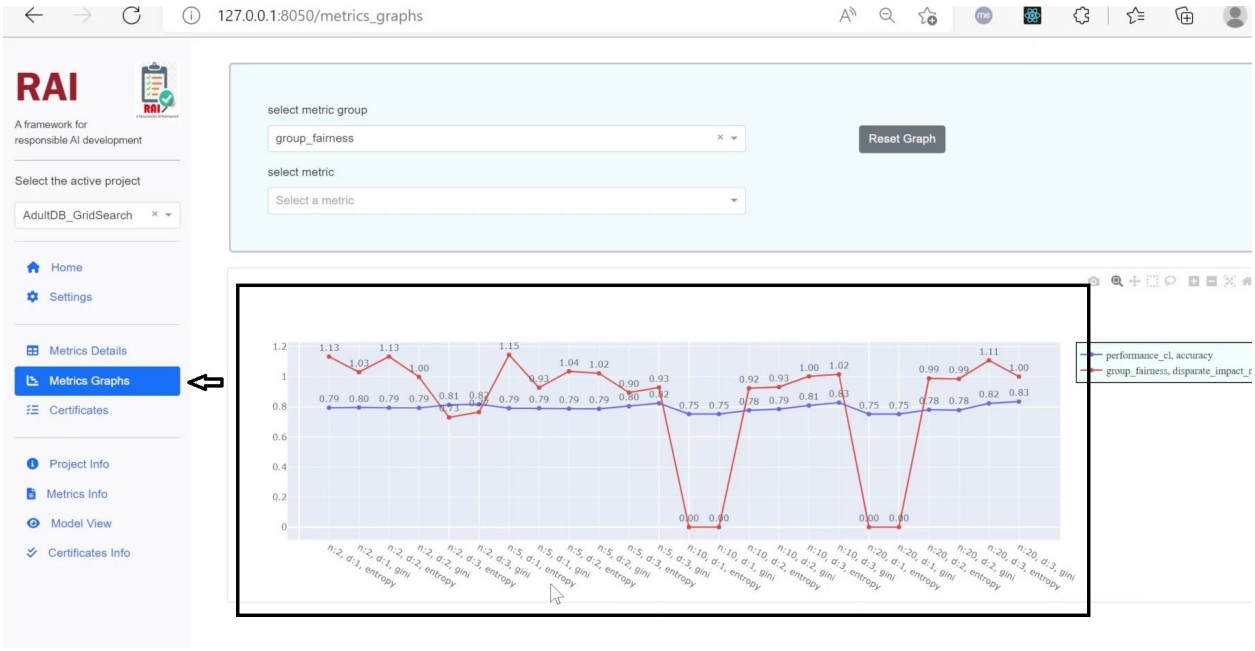
- Rai can help us with the Model selection
- We can select a Project here

Select project



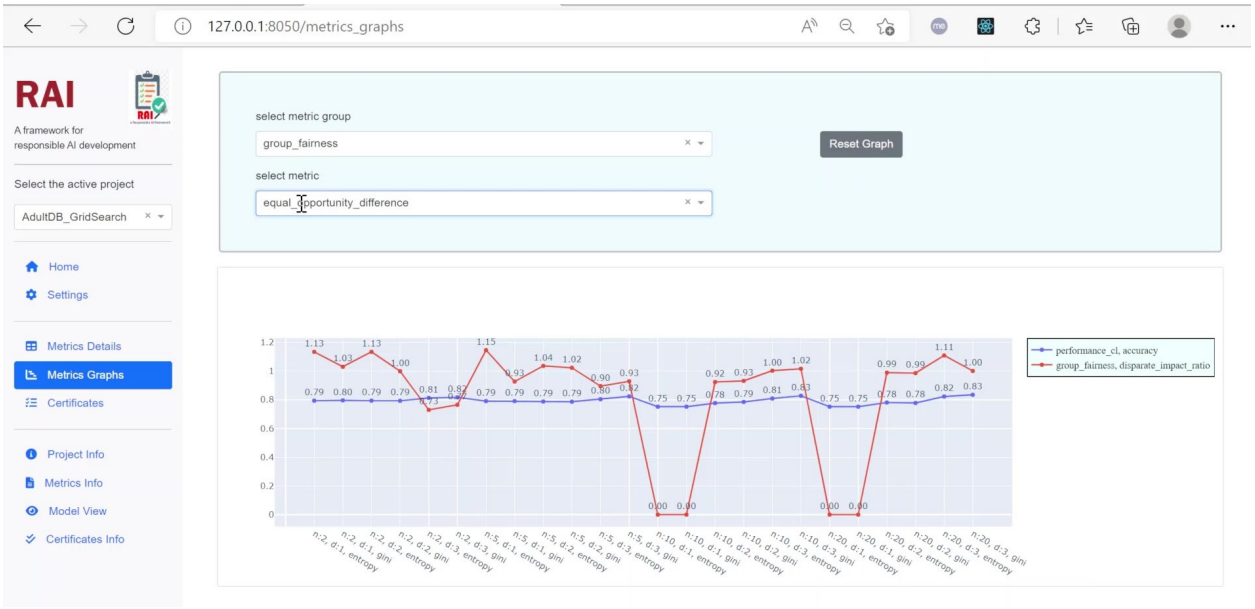
- We can go to Metric Graphs
- Metric Graphs show here how individual parameters and metrics have changed during model development

Metric graph



- Here, for instance, we have performed some Grid searches to select the best model for the task
- We can show individual metrics of interest

Metric performance



- Monitor how the system is performing in each individual case
- This helps us to select the best model that fits our desired characteristics

Individual case

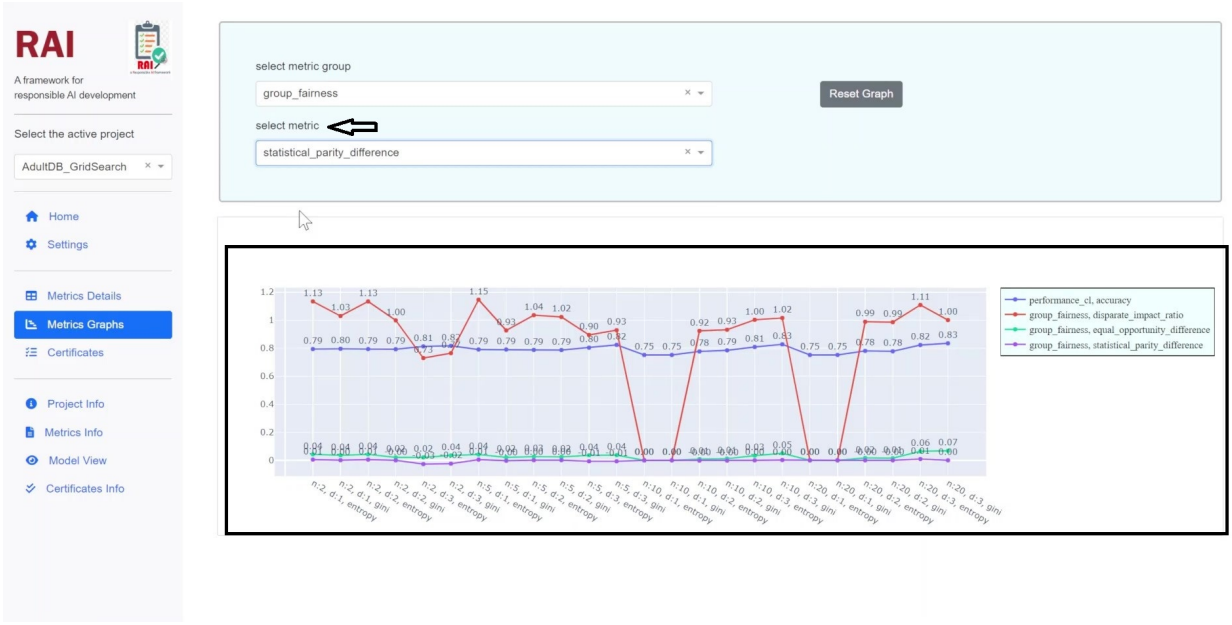


Fig. 1: Model_selection

Important: Through RAI we can detect it before it becomes a problem or respond to it when it arises by putting the right systems in place early and staying on top of data collection, labeling, and implementation.

CHAPTER

SEVEN

DEMO CASES

Important: Demo projects will show some of the capabilities of RAI.

- **Run the demos by using.**

```
``python demo_filename`` for instance
```

FileName	Description
demos> python .adult_demo_grid_search.py	This demo uses the Adults dataset (https://archive.ics.uci.edu/ml/datasets/adult) to show how RAI can be used in model selection.
demos> python .image_class_analysis.py	This demo uses Cifar10 dataset and shows how RAI can be used to evaluate image classification tasks.
demos> python .adult_demo_custom_metrics.p	This demo shows how RAI can be used to create dynamic custom metrics to evaluate your model and display them on the dashboard.
demos> python .image_class_training.py	This demo uses Cifar10 dataset and shows how RAI can be used monitor image processing tasks during training.
demos> python .tabular_class_console.py	This demo shows how RAI can be used without the dashboard to calculate and report on the metrics for a machine learning task.
demos> python .text.py	This demo show how RAI and its dashboard can be used for evaluating the natural language modeling tasks.
demos> python .text_output.py	This demo show how RAI and its dashboard can be used for evaluating the natural language modeling tasks.

CHAPTER

EIGHT

DASHBOARD

The goal is to provide a generic framework to perform various types of analysis, from data quality assessment to model selection based on performance, fairness, and robustness criteria.

8.1 RAI Dashboard features

- RAI dashboards display metrics and critical data and give the user a visual representation of them.
- The RAI dashboard computes all metrics and visualizes the currently running code in real time.
- RAI dashboard can handle a wide variety of models and store information for each model, from text-based models to image-based models.
- RAI dashboard automatically determines what metrics are relevant by examining the type of model, data, and task provided by the user.
- With RAI, users can view data at a high level by scanning dashboards like input and output visualizations.

8.2 How To Run RAI Dashboard

- When the python file `main.py` is ran, Dash is running on localhost and the port is 8050, if we follow and click on the URL or copy the URL and paste in the browser, you can access the dashboard.

Description: Run the main.py by

```
 Dashboard> python .\main.py

-Click the following link to access Dashboard.
-Dash is running on ...
```

Example:

8.3 Interaction of RAI Dashboard

How it links to RAI project (via sqlite..)

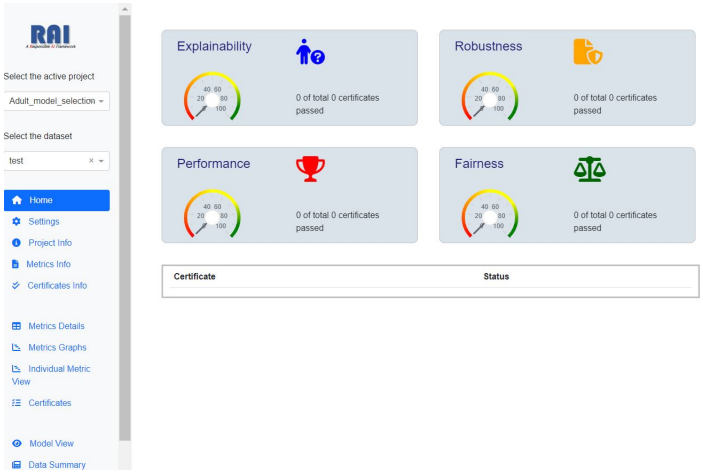


Fig. 1: RAI Dashboard Homepage.

How it shows the Metrics, Certificates and Analysis page

- **Metrics:**

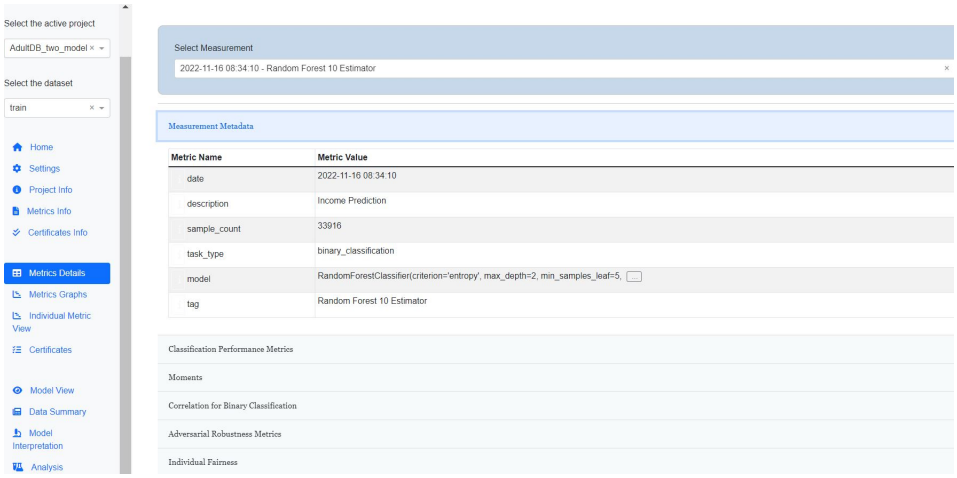


Fig. 2: Metric Details Page.

- **Certificates:**
- **Analysis:**

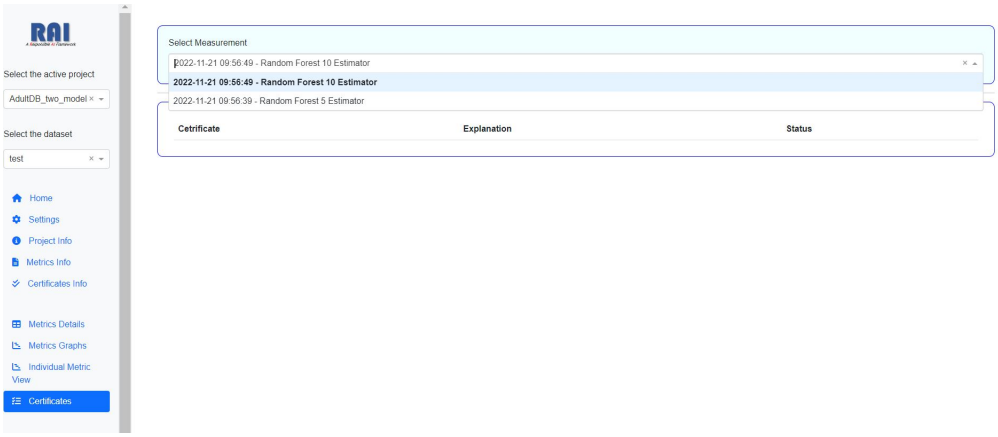


Fig. 3: Certificate Page.

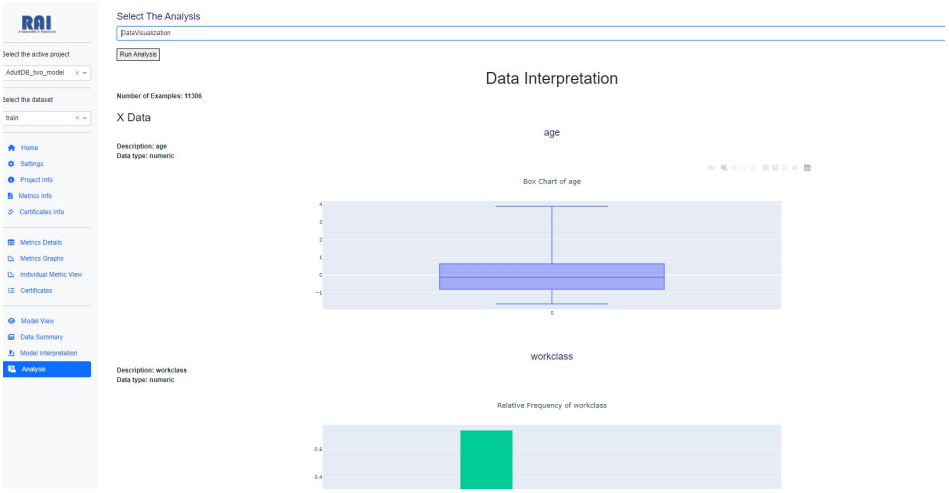


Fig. 4: Analysis Page.

CHAPTER

NINE

BASIC COMPONENT OF RAI DESIGN

9.1 AISystem

- **Representation**
- AISystems are the main class users interact with in RAI, they capture key information about an AI.
- This information is passed during construction and includes a name, a task type, a MetaDatabase, a Dataset and a Model.
- **Interaction**
- AISystems make it simple to run computations and get metric values, and are needed to run an Analysis and run Certifications.
- After making an AISystem, users can use the compute function to generate all relevant metrics related to their model and dataset.
- The Model, Task Type and MetaDatabase are RAI classes which provide critical information to the AISystem, allowing it to determine which metrics and analyses are relevant.
- After computing metrics, users can get retrieve metric values using the get_metric_values function.
- When provided a network's functions to generate predictions or values, AISystems can use models and run evaluations without requiring user involvement.
- You can provide custom expressions or functions as metrics that will be evaluated for every measurement.

Example:

AI_sys file example

```
def initialize(self, user_config: dict = {},
               custom_certificate_location: str = None,
               custom_metrics: dict = {},
               custom_functions: list = None
               ):
    """
    :param user_config: Takes user config as a dict
    :param custom_certificate_location: certificate path by default it is None
    :param custom_metrics: dict of custom metrics you want to display on the dashboard
    :param custom_functions: list of custom functions that take the existing metrics as input and return a value

    :return: None
    """
    self.user_config = user_config
    self.custom_metrics = custom_metrics
    self.custom_functions = custom_functions
    masks = {"scalar": self.meta_database.scalar_mask, "categorical": self.meta_database.categorical_mask,
             "image": self.meta_database.image_mask, "text": self.meta_database.text_mask}
    self.dataset.separate_data(masks)
    self.meta_database.initialize_requirements(self.dataset, "fairness" in user_config)
    self.metric_manager = MetricManager(self)
    self.certificate_manager = CertificateManager()
    self.certificate_manager.load_stock_certificates()
    if custom_certificate_location is not None:
        self.certificate_manager.load_custom_certificates(custom_certificate_location)
```

Important: Rai utilization: RAI will utilize AISystem to compute and determine which metrics are relevant (e.g. of the chosen model design).

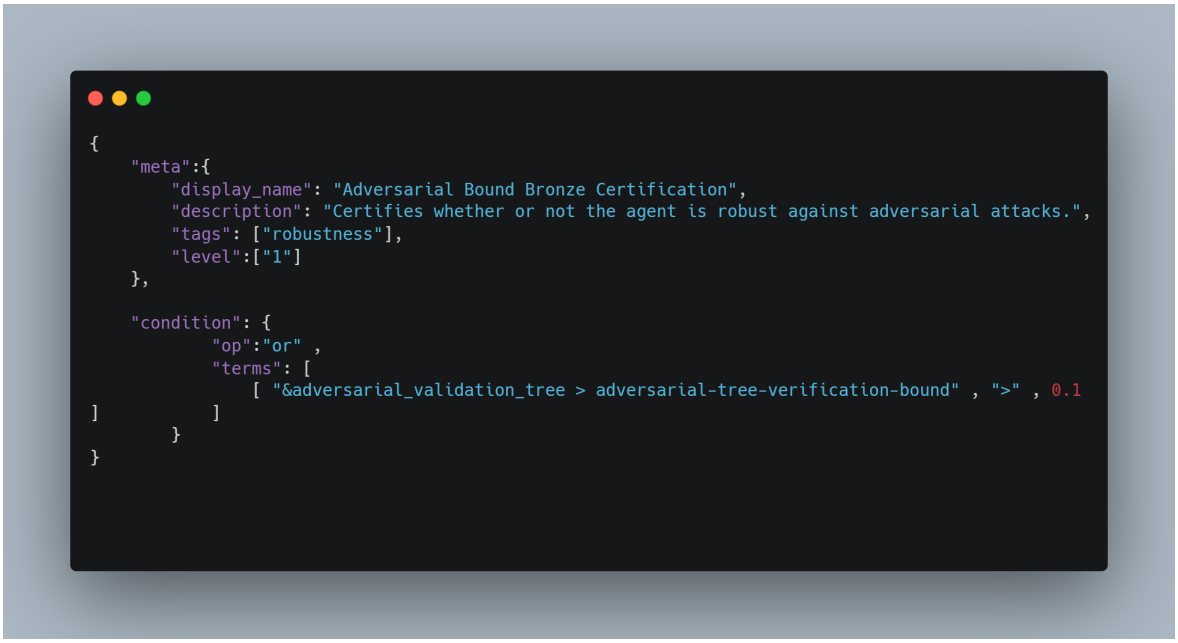
9.2 Certificates

- **Representation**
- Certificates allow users to quickly and easily define and communicate standards for AISystems in different domains and tasks.
- Once a certificate has been added to an AISystem, the AISystem can quickly and easily evaluate whether or not it meets the standards of the certificate allowing for quick yet robust evaluation.
- Certificates are written in JSON and can contain logical and relational operators, with the ability to retrieve any metric associated with an AISystem.
- **Interaction**
- Custom Certificates can be added to an AISystem by passing in a filepath to the certificate file while initializing the AISystem.
- Certificates will be evaluated when the AISystem calls its compute function.
- Certificates can be retrieved by calling the get_certificate_values function on the AISystem.

Example:

- **display_name**: Adversarial Bound Bronze Certification.
- **description**: Certifies whether or not the agent is robust against adversarial attacks.

Certi file example



```
{
    "meta":{
        "display_name": "Adversarial Bound Bronze Certification",
        "description": "Certifies whether or not the agent is robust against adversarial attacks.",
        "tags": ["robustness"],
        "level":["1"]
    },

    "condition": {
            "op":"or" ,
            "terms": [
                [ "&adversarial_validation_tree > adversarial-tree-verification-bound" , ">" , 0.1
]           ]
        }
}
```

Important: **Rai utilization:** RAI will carry out detailed analyses (e.g. of the chosen model design) and tests (e.g. robustness, bias, explainability) and define a certification that have Accuracy features.

9.3 Metrics

- **Representation**
- Each Metric comes with metadata, including its name and description, as well as a function to compute the metric.
- Metrics are grouped into MetricGroups, which are collections of Metrics with similar compatibility and functionality.
- AISystems access metrics through MetricManagers which are responsible for checking compatibility between MetricGroups and AISystems, as well as computing and retrieving specific Metric values.
- MetricManagers are automatically created and managed by AISystems and are the key to running Metrics and retrieving their values.
- **Interaction**
- Interaction with Metrics are done through MetricManagers.
- MetricManagers are capable of quickly finding all MetricGroups compatible with an AISystem.
- RAI ensures that dependencies between Metrics are satisfied with no circular dependency issues.
- Functionality is provided to search for specific Metrics based on Metric Name, Metric Group Name, Category, and Tags.
- Metrics are compatible with both whole and batched data.

Example:

Metric file example

```
def get_metric_values(self):
        results = {}
        for metric_name in self.metrics:
            if self.metrics[metric_name].type == 'vector':
                results[metric_name + "-single"] = self.metrics[metric_name].value[0]
                val = self.metrics[metric_name].value[1]
                if type(self.metrics[metric_name].value[1]) is np.ndarray:
                    val = val.tolist()
                results[metric_name + "-individual"] = val  # Easily modify to export for each
value.      elif self.metrics[metric_name].type == 'feature-array':
                val = val = self.metrics[metric_name].value
                if type(self.metrics[metric_name].value) is np.ndarray:
                    val = val.tolist()
                results[metric_name] = val
            else:
                results[metric_name] = self.metrics[metric_name].value
        return results
```

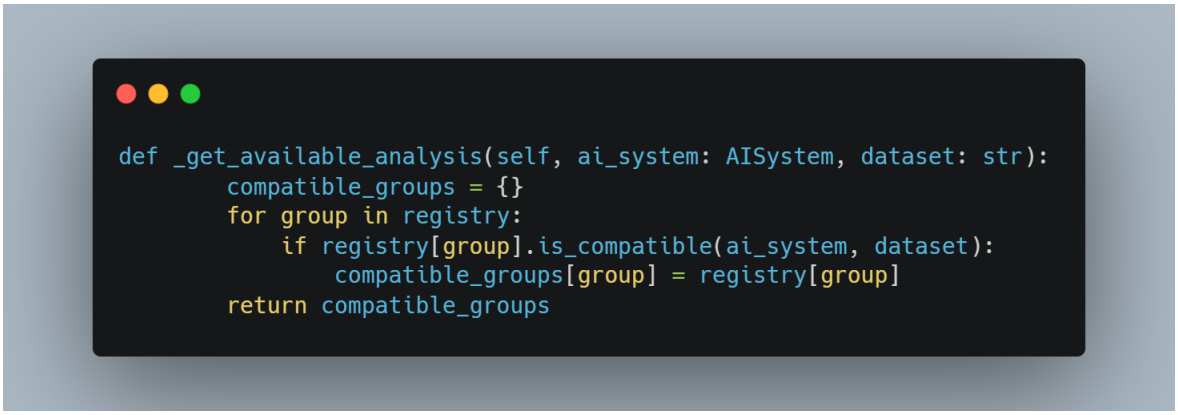
Important: **Rai utilization:** RAI will utilize Metrics to monitor and measures the performance of a model (during training and testing).

9.4 Analysis

- **Representation**
- While metrics are typically general and simple to calculate, Analyses are finegrained evaluations to run on specific AISystems.
- Analyses provide a way for users to quickly and easily run complex experiments compatible with their model, with built in visualizations.
- Analyses are easy to create allowing users to quickly and easily make their own custom Analyses for their specific needs using any attribute of the AISystem.
- **Interaction**
- Analyses are managed by the AnalysisManger and are given access to the AISystem and Dataset through the RaiDB class.
- Similar to MetricManagers, AnalysisManagers check compatibility between Analyses and AISystems and handle the computation of Analyses.
- Running specific analyses is done through the run_analysis function.

Example:

Analysis file example



```
def _get_available_analysis(self, ai_system: AISystem, dataset: str):
        compatible_groups = {}
        for group in registry:
            if registry[group].is_compatible(ai_system, dataset):
                compatible_groups[group] = registry[group]
        return compatible_groups
```

Important: **Rai utilization:** RAI will carry out detailed analyses and automates report generation and makes data easy to understand.

CHAPTER

TEN

CONTRIBUTE AND EXTEND RAI

- **How to contribute and extend RAI (User Guide)**

Here's the short summary

- If you are a first-time contributor,
- Go to ResponsibleAI and click the "fork" button to create your own copy of the project from *master*.
- Clone the project to your local computer.
- Develop your contribution.
- Push your changes back to your fork on GitHub.

10.1 Adding Metric Group

What are its requirements

- To add a New metrics, we need to create 3 files inside Metrics folder.
- **__init__.py file**.
- **json file**.
- **Python file**.
- In `__init__.py` file we have to import the python file that we have created inside the folder.

  Example:

__init__.py file example

How to expand RAI using Metric group

- Inside `json file`, we need to define parametrs as name, display_name, compatibility, dependency_list, tags, complexity_class, metrics.

  parameter
  name:

  parameter
  display_name:

parameter
compatibility:

parameter
dependency_list:

parameter
tags:

parameter
complexity_class:

parameter
metrics:

Example:

---

json_metric file example

```
{
    "name": "basic_explainablity",
    "display_name": "Basic Explainability",
    "compatibility": {"task_type": [],
                      "data_type": ["numeric"],
                      "output_requirements": [],
                      "dataset_requirements": [],
                      "data_requirements": ["NumpyData"]},
    "dependency_list": [],
    "tags": ["robustness", "Normalization"],
    "complexity_class": "linear",
    "metrics": {
        "explainable_model": {
            "display_name": "explainable model",
            "type": "Boolean",
            "has_range": false,
            "range": [null, null],
            "explanation": "Placeholder method for if a method is explainable.",
            "citation": ""
        }
    }
}
```

Create subclass of class and implement the method

- We can create subclass of class and implement the methods.

- In `python file` , we need to create a class for respective metric_group and we need to define methods for update and compute inside the class.

Example:

---

metric_python_file example

```
class GeneralDatasetFairnessGroup(MetricGroup, class_location=os.path.abspath(__file__)):
    def __init__(self, ai_system) -> None:
        super().__init__(ai_system)

    @classmethod
    def is_compatible(cls, ai_system):
        compatible = super().is_compatible(ai_system)
        return compatible \
            and "fairness" in ai_system.metric_manager.user_config \
            and "protected_attributes" in ai_system.metric_manager.user_config["fairness"] \
            and len(ai_system.metric_manager.user_config["fairness"]["protected_attributes"]) >
0
    def update(self, data):
        pass

    def getConfig(self):
        return self.config

    def compute(self, data_dict):
        data = data_dict["data"]
```

10.2 Adding Certificates

What are is requirements

- To add a New Certificates , we need to create a json file inside a standard folder.

How to expand RAI using Certificates

- For certificates there are two key value pairs , first one meta and second one conditions.
- Inside meta we need to give diplay name, description, tags and level.
- Inside condition we need to give operator and terms.

Create a certificate and implement

- Inside certificate folder, go to standard folder and their make a `json file` and fill all given parameters.

  parameter
  meta:

  parameter
  condition:

Example:

sample certficate

```
{
    "meta":{
        "display_name": "Adversarial Bound Bronze Certification",
        "description": "Certifies whether or not the agent is robust against adversarial attacks.",
        "tags": ["robustness"],
        "level":["1"]
    },

    "condition": {
            "op":"or" ,
            "terms": [
                [ "&adversarial_validation_tree > adversarial-tree-verification-bound" , ">" , 0.1
]           ]
        }
}
```

10.3 Adding Analysis

What are is requirements

- To add a New Analysis, we need to create 3 files inside Analysis folder.
- **__init__.py file**.
- **json file**.
- **Python file**.
- In `__init__.py` file - we have to import the py file that we have created inside the folder.

Example:

__init__.py file example

How to expand RAI using Analysis

- Inside `json file`, we need to give parametrs name, display_name, compatibility, src, dependency_list, tags, complexity_class.

  parameter
  name:

  parameter
  display_name:

parameter
compatibility:

parameter
src:

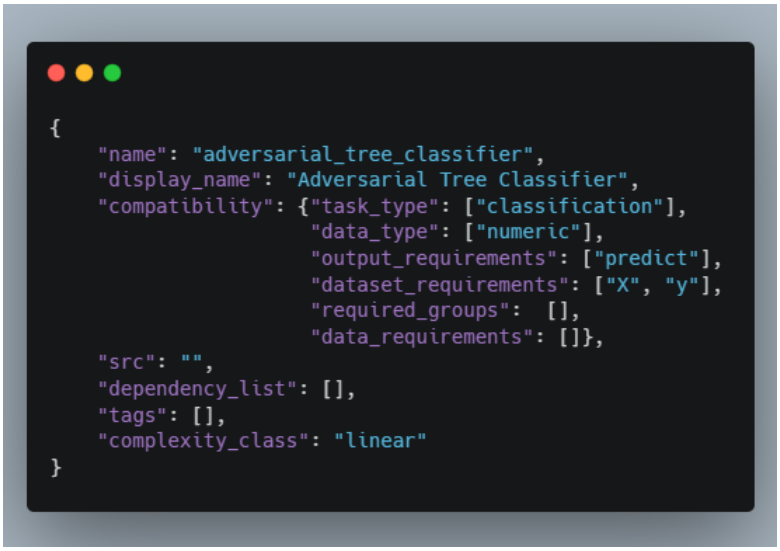
parameter
dependency_list:

parameter
tags:

parameter
complexity_class:

Example:

Analysis_json_file example



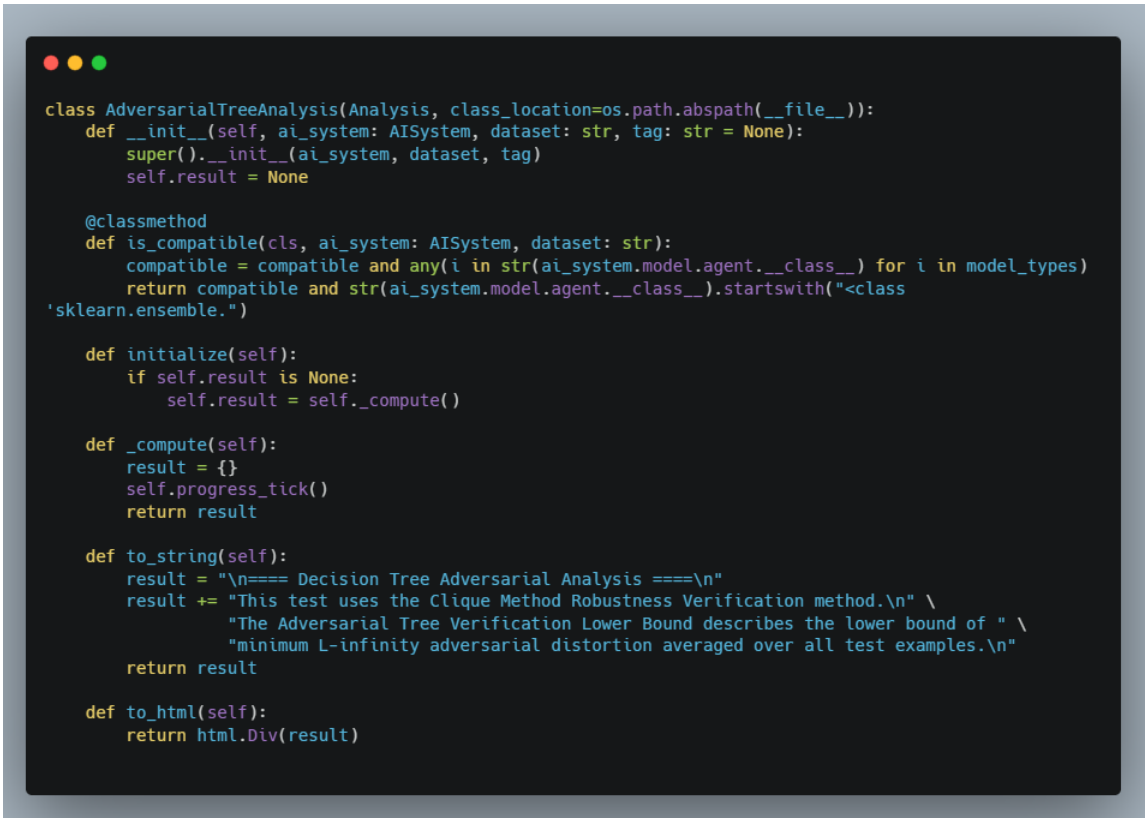
```
{
    "name": "adversarial_tree_classifier",
    "display_name": "Adversarial Tree Classifier",
    "compatibility": {"task_type": ["classification"],
                      "data_type": ["numeric"],
                      "output_requirements": ["predict"],
                      "dataset_requirements": ["X", "y"],
                      "required_groups":  [],
                      "data_requirements": []},
    "src": "",
    "dependency_list": [],
    "tags": [],
    "complexity_class": "linear"
}
```

Create subclass of class and implement the method

- We can create subclass of class and implement the methods.
- In `python file` , we need to create a class for respective Analysis and we need to define methods for initialize ,compute, to_string, to_html inside the class.

Example:

Analysispyfile.png example



```
class AdversarialTreeAnalysis(Analysis, class_location=os.path.abspath(__file__)):
    def __init__(self, ai_system: AISystem, dataset: str, tag: str = None):
        super().__init__(ai_system, dataset, tag)
        self.result = None

    @classmethod
    def is_compatible(cls, ai_system: AISystem, dataset: str):
        compatible = compatible and any(i in str(ai_system.model.agent.__class__) for i in model_types)
        return compatible and str(ai_system.model.agent.__class__).startswith("<class
'sklearn.ensemble.")

    def initialize(self):
        if self.result is None:
            self.result = self._compute()

    def _compute(self):
        result = {}
        self.progress_tick()
        return result

    def to_string(self):
        result = "\n==== Decision Tree Adversarial Analysis ====\n"
        result += "This test uses the Clique Method Robustness Verification method.\n" \
                  "The Adversarial Tree Verification Lower Bound describes the lower bound of " \
                  "minimum L-infinity adversarial distortion averaged over all test examples.\n"
        return result

    def to_html(self):
        return html.Div(result)
```

CHAPTER

ELEVEN

CONTRIBUTION OF USERS TO EXPAND ITS FEATURES

11.1 Contributing to RAI

Thank you for taking time to start contributing! We want to make contributing to this project as easy and transparent as possible, whether it's:

- Reporting a bug
- Discussing the current state of the code
- Submitting a fix
- Proposing new features
- Becoming a maintainer

We Develop with Github

We use github to host code, to track issues and feature requests, as well as accept pull requests.

Pull requests are the best way to propose changes to the codebase. We actively welcome your pull requests:

1. Fork the repo and create your branch from *master*.
2. If you've added code that should be tested, add tests.
3. If you've changed APIs, update the documentation.
4. Ensure the test suite passes.
5. Make sure your code lints.
6. Issue that pull request!

Any contributions you make will be under the Apache License, Version 2

- In short, when you submit code changes, your submissions are understood to be under the same Apache License that covers the project.
- Feel free to contact the maintainers if that's a concern.

11.2 Report bugs using Github's Issues

- We use GitHub issues to track public bugs. Report a bug by opening a new issue
- Write bug reports with detail, background, and sample code.

Please consider to include the following in a bug report:

- A quick summary and/or background
- Steps to reproduce - Be specific! - Give sample code if you can.
- What you expected would happen
- What actually happened
- Notes (possibly including why you think this might be happening, or stuff you tried that didn't work)

License

- By contributing, you agree that your contributions will be licensed under its Apache License, Version 2.

References

- This document was adapted from here.

CHAPTER

TWELVE

RAI

12.1 RAI.AISystem

12.1.1 RAI.AISystem.ai_system module

class RAI.AISystem.ai_system.**AISystem**(*name: str*, *task: str*, *meta_database: MetaDatabase*, *dataset: Dataset*, *model:* Model, *enable_certificates: bool = True*)

Bases: `object`

AI Systems are the main class users interact with in RAI. When constructed, AISystems are passed a name, a task type, a MetaDatabase, a Dataset and a Model.

Parameters

- **name** – Create a new string object from the given object. If encoding or errors is specified, then the object must expose a data buffer that will be decoded using the given encoding and error handler. Otherwise, returns the result of object.__str__() (if defined) or repr(object). encoding defaults to sys.getdefaultencoding(). errors defaults to 'strict'.
- **task** – Create a new string object from the given object. If encoding or errors is specified, then the object must expose a data buffer that will be decoded using the given encoding and error handler. Otherwise, returns the result of object.__str__() (if defined) or repr(object). encoding defaults to sys.getdefaultencoding(). errors defaults to 'strict'.
- **meta_database** – The RAI MetaDatabase class holds Meta information about the Dataset. It includes information about the features, and contains maps and masks to quick get access to the different feature data of different information.
- **dataset** – The RAI Dataset class holds a dictionary of RAI Data classes, for example {'train': trainData, 'test': testData}, where trainData and testData are RAI Data objects.
- **model** – Model is RAIs abstraction for the ML Model performing inferences. When constructed, models are optionally passed the name, the models functions for inferences, its name, the model, its optimizer, its loss function, its class and a description. Attributes of the model are used to determine which metrics are relevant.
- **enable_certificates** – Returns True when the argument x is true, False otherwise. The builtins True and False are the only two instances of the class bool. The class bool is a subclass of the class int, and cannot be subclassed.

add_certificates()

Add certificates values to the existing metrics :return: None

add_custom_metrics()

Add custom metrics to existing metrics

Returns
None

compute(*predictions: dict*, *tag=None*) → None

Compute will tell RAI to compute metric values across each dataset which predictions were made on

Parameters

- **predictions(dict)** – Prediction value from the classifier
- **tag** – by default None

Returns
None

display_metric_values(*display_detailed: bool = False*)

Parameters
display_detailed – if True we need to display metric explanation if False we don't have to display

Returns
None

Displays the metric values

get_certificate_info()

Returns the metadata of the certificate_manager class

Parameters
self – None

Returns
Certificate info from certificate_manager

get_certificate_values() → dict

Returns the last used certificate information

Parameters
self – None

Returns
Certificate infomation(dict) Returns the last used certificate information

get_data(*data_type: str*) → Data

get_data accepts data_type and returns the data object information

Parameters
data_type(str) – Get the data type information

Returns
Dataset datatype information(str)

get_data_summary() → dict

process the data and returns the summary consisting of prediction, label details

Parameters
self – None

Returns
Data Summary(Dict)

`get_metric_info()`

Returns the metadata of the metric_manager class

Parameters
self – None

Returns
metric Manager metadata

`get_metric_values()` → dict

Returns the last metric values in the form of key value pair

Parameters
self – None

Returns
last metric values(dict)

`get_project_info()` → dict

Fetch the project information like name, configuration, metric user config and Returns the project details

Parameters
self – None

Returns
Project details(dict)

`initialize`(*user_config: dict = {}*, *custom_certificate_location: str | None = None*, *custom_metrics: dict = {}*, *custom_functions: list | None = None*)

Parameters

- **user_config** – Takes user config as a dict
- **custom_certificate_location** – certificate path by default it is None
- **custom_metrics** – dict of custom metrics you want to display on the dashboard
- **custom_functions** – list of custom functions that take the existing metrics as input and return a value

Returns
None

`run_compute`(*tag=None*) → None

Run Compute automatically generates outputs from the model, and compute metrics based on those outputs

Parameters
tag – tag by default None or we can pass model as a string

Returns
Data Summary(Dict)

`update`(*data*)

12.1.2 RAI.AISystem.model module

class RAI.AISystem.model.**Model**(*output_features=None*, *predict_fun=None*, *predict_prob_fun=None*, *generate_text_fun=None*, *generate_image_fun=None*, *name=None*, *display_name=None*, *agent=None*, *loss_function=None*, *optimizer=None*, *model_class=None*, *description=None*)

Bases: object

Model is RAIs abstraction for the ML Model performing inferences. When constructed, models are optionally passed the name, the models functions for inferences, its name, the model, its optimizer, its loss function, its class and a description. Attributes of the model are used to determine which metrics are relevant.

12.2 RAI.Analysis

12.2.1 Submodules

12.2.2 RAI.Analysis.analysis module

class RAI.Analysis.analysis.**Analysis**(*ai_system: AISystem*, *dataset: str*, *tag: str | None = None*)

Bases: ABC

abstract initialize()

classmethod is_compatible(*ai_system*, *dataset: str*)

Parameters

- **ai_system** – input the ai_system object
- **dataset** – input the dataset

Returns
class object

Returns the classifier and sklearn object data

progress_percent(*percentage_complete*)

Parameter
percentage_complete

Returns
None

Shows the progress percent value

progress_tick()

Parameter
None

Returns
None

On every compute it changes the current_tick value

set_connection(*connection*)

Parameters
connection – inputs connection data

Returns
None

Connection is a function that accepts progress, and pings the dashboard

abstract to_html()

abstract to_string()

12.2.3 RAI.Analysis.analysis_manager module

class RAI.Analysis.analysis_manager.**AnalysisManager**

Bases: object

get_available_analysis(*ai_system: AISystem*, *dataset: str*)

Parameters

- **AISystem** – input the ai_system obj
- **dataset** – input the dataset

Returns
list.

Returns the lists of analysis data

run_all(*ai_system: AISystem*, *dataset: str*, *tag: str*)

Parameters

- **AISystem** – input the ai_system obj
- **dataset** – input the dataset
- **tag** – By default None else given tag Name

Returns
None.

Returns the analysis data result analysis

run_analysis(*ai_system: AISystem*, *dataset: str*, *analysis_names*, *tag=None*, *connection=None*)

Parameters

- **AISystem** – input the ai_system obj
- **dataset** – input the dataset
- **tag** – By default None else given tag Name
- **analysis_names** – analysis_names data set
- **connection** – By default None

Returns
Dict.

Returns the analysis data result analysis

12.2.4 RAI.Analysis.analysis_registry module

RAI.Analysis.analysis_registry.**register_class**(*class_name*, *class_object*)

Parameters

- **class_name** – inputs the name for the class and that should be unique
- **class_object** – class object is given as the input

Returns
registered data in the form of dictionary containing class name as the key and class object as the value

12.3 RAI.certificates

12.3.1 Submodules

12.3.2 RAI.certificates.certificate module

class RAI.certificates.certificate.**Certificate**

Bases: object

Certificate Objects contain information about a particular certificate. Certificates are automatically loaded in by CertificateManagers and perform evaluation using metric data they are provided in combination with the certificate data loaded in.

evaluate(*metrics*, *certs*)

From the certificate json file condition key is selected and based on that evalutions will happen

Parameters

- **metrics** – metrics data is provided based on that evaluation will happen
- **certs** – certificate data is provided based on that evaluation will happen

Returns
Returns the evaluation result based on the input data

load_from_json(*json_file*)

opens the certificate json file and load all the information

Parameters
json_file – json_file file path of the certificate json file is shared

Returns
None

12.3.3 RAI.certificates.certificate_manager module

class RAI.certificates.certificate_manager.**CertificateManager**

Bases: object

CertificateManager is a class automatically created by AISystems. This class loads a file containing information on which certificates to use, before creating associated Certificate Objects, as well as prompting their associated evaluations.

compute(*metric_values*)

Accepts the metric values and returns the value as per the name of the certificate

Parameters
metric_values(dict)

Returns
metric results(list)

get_metadata() → dict

return the certificate metadata information

Parameters
self – None

Returns
metadata(dict)

load_custom_certificates(*filename*)

Loads all certificates found in a custom filepath

Parameters
filename – where we need to get the details

Returns
None

load_stock_certificates()

Loads all certificates found in the stock certificate file

Parameters
self – None

Returns
None

12.4 RAI.metrics

12.4.1 Submodules

12.4.2 RAI.metrics.metric module

class RAI.metrics.metric.**Metric**(*name*, *config*)

Bases: object

Metric class loads in information about a Metric as part of a Metric Group. Metrics are automatically created by Metric Groups.

load_config(*config*)

loads the config details consisting of tags, has_range, range, explanation, type and display_name

Parameters
config – Config details

Returns
None

12.4.3 RAI.metrics.metric_group module

class RAI.metrics.metric_group.**MetricGroup**(*ai_system*)

Bases: ABC

MetricGroups are a group of related metrics. This class loads in information about a metric group from its .json file. This class then creates associated metrics for the group, provides compatibility checking, run computes. Metric Groups are created by MetricManagers.

abstract compute(*data*)

compute_batch(*data*)

config = None

create_metrics(*metrics_config*)

Create the metric and assign name and tags to it

Param
metrics_config

Returns
None

export_metric_values()

Returns the metric with the name and its corresponding value

Param
None

Returns
dict

finalize_batch_compute()

get_metric_values()

Returns the metric with the name and its corresponding value

Param
None

Returns
dict

classmethod is_compatible(*ai_system*)

Checks if the group is compatible with the provided AiSystem

Param
ai_system

Returns
Compatible object

`load_config`(*config*)

Fetch the metric data from config

Param
config

Returns
Boolean

`name = ''`

`reset`()

Reset the status

Param
None

Returns
None

`update`(*data*)

12.4.4 RAI.metrics.metric_manager module

`class RAI.metrics.metric_manager.MetricManager`(*ai_system*)

Bases: `object`

MetricManager is used to create and Manage various MetricGroups which are compatible with the AISystem. MetricManager is created by the AISystem, and will load in all available MetricGroups compatible with the AISystem. MetricManager also provides functions to run computes for all metric groups, get metadata about metric groups, and get metric values.

`compute`(*data_dict*) → dict

Perform computation on metric objects and returns the value as a metric group in dict format

Parameters
data_dict – Accepts the data dict metric object

Returns
returns the value as a metric group

`get_metadata`() → dict

Return the metric group metadata information

Parameters
self – None

Returns
dict-Metadata

`get_metric_info_flat`() → dict

Returns the metric info

Parameters
self – None

Returns
Returns the metric info data in dict

initialize(*user_config: dict | None = None*, *metric_groups: List[str] | None = None*, *max_complexity: str = 'linear'*)

Find all compatible metric groups and Remove metrics with missing dependencies and Check for circular dependencies

Parameters

- **user_config(dict)** – user config data
- **metric_groups** – metric groups data as a list
- **max_complexity** – default linear

Returns
None

iterator_compute(*data_dict*, *preds: dict*) → dict

Accepts data_dict and preds as a input and returns the metric objects from a batch of metric group

Parameters

- **data_dict** – Accepts the data dict metric object
- **preds** – prediction value from the detection

Returns
returns the metric objects from a batch of metric group

reset_measurements() → None

Reset all the certificate, metric, sample and time_stamp values

Parameters
self – None

Returns
None

search(*query: str*) → dict

Searches all metrics.Queries based on Metric Name, Metric Group Name, Category, and Tags

Parameters
query – query(str) group data information as input

Returns
Returns the search results based on the query

standardize_user_config(*user_config: dict*)

Accepts user config values and make in standard group

Parameters
user_config(dict) – user config data

Returns
None

12.4.5 RAI.metrics.metric_registry module

Registers Metric Classes on creation. All valid metric groups can then be found in the registry dictionary.

RAI.metrics.metric_registry.**register_class**(*class_name*, *class_object*)

PYTHON MODULE INDEX

r

INDEX