# Dokumentation für das Decision Boundary Code Projekt

## Installation der Bibliotheken

Mit der „requirements.txt“ Datei kann eine Anaconda Umgebung mit den nötigen Bibliotheken erstellt werden. Die Liste der Packages ist nicht ganz minimal. Überprüft nach erstellen der Umgebung noch einmal ob tatsächlich Python 3.10 verwendet wird. Bei korrekter Verwendung der „requirements.txt“ ist das automatisch der Fall.

## Ordner „datasets“

Im Ordner „datasets“ werden die Datensätze und dazugehörigen Modelle/neuronalen Netzwerke gespeichert. Die Nomenklatur (z.B. „model\_{Datensatz}.pt“, „{Datensatz}\_data\_test.npy“,…), die Ordnerstruktur, und der Dateityp sind festgelegt und sollten konsistent so beibehalten werden, da der Code so auf diese zugreift. Die Datensätze „Iris“ und „Fetal Health“ liegen (mit bereits trainierten Modellen) bereits in passender Form vor.

## Ordner „model training“

Im Ordner „model training“ liegen Skripte zum Training von neuen neuronalen Netzen (z.B. „train\_iris.py“), welche das Modell gleich im richtigen Format abspeichern. Hierbei wird im Wesentlichen auf die Klasse „Trainer“ aus „Training\_NN.py“ zugegriffen. Außerdem wichtig: Die Konfiguration (Anzahl Layer, Neuronen, Aktivierungsfunktionen) des Netzes muss im Skript „./utils/models.py“ vorher spezifiziert und dann in „Training\_NN.py“ und „train\_{Datensatz}.py“ ausgewählt werden. Das gegebene „train\_iris.py“ kann hier gut zum Verständnis beitragen.

## “Width” und “Certainty Vector”

“width” und “certainty\_vector” definieren im Code Projekt wichtige Variablen:

Da immer der Bereich des Inputraums (inkl. DBs, andere Instanzen, …) um eine bestimmte (zu erklärende) Dateninstanz herum betrachtet wird, muss die Größe dieses „würfelförmigen“ Bereichs definiert werden. Width legt dafür die Breite des Würfels in jeder Dimension fest, d.h. der Raum in Weite width/2 wird in jeder Richtung links und rechts vom Wert der Dateninstanz betrachtet und kann visualisiert werden.

Die Variable/Liste Certainty Vector ermöglicht die Einschränkung des Inputraums auf einen Unterraum niedrigerer Dimension. In den gewünschten Dimensionen des Unterraums steht im entsprechenden Eintrag der Liste der Wert „None“, in den Dimensionen, die im Unterraum nicht betrachtet werden sollen, ein fester reeller Wert. Falls z.B. der 2 dimensionale Unterraum der Dimensionen 1 und 3 um die Instanz [1, 2, 3, 1] betrachtet werden soll, lautet der certainty vector [None, 2, None ,1]. Dieser wird bei der Berechnung der DBs übergeben.

## Skript „decision\_boundaries\_v2.py“

Das Skript “decision\_boundaries\_v2.py” ist das für euch interessanteste Hauptskript. Hier findet sich die Python Klasse „Decisionboundaries“ welche die zu visualisierenden Entscheidungsgrenzen enthält.

Im „main“ Teil des Skriptes befindet sich ein Beispiel Code, welcher die wichtigsten Funktionen durchführt. Diesen könnt ihr testen und schrittweise Änderungen hinzufügen um den Code zu verstehen! Außerdem enthält er schon eine recht grundlegende Visualisierung welche ihr als Grundlange in Betracht ziehen könnt, aber definitiv nicht verwenden müsst!

Eine kurze Beschreibung der jeweiligen Funktionen findet ihr kommentiert im Skript.

HINWEIS: Generell ist zu beachten, dass aus methodischen Gründen immer folgende Reihenfolge einzuhalten ist:

Initialisieren des DecisionBoundaries Objektes => „getLinearRegions“ => „getDecisionBoundariesExact“ => „getDistancestoallBoundariesOpt“ oder „plotExactDecisionBoundaries“.

Jede dieser Funktionen kann nur nach der vorherigen ausgeführt werden und führt sonst zu Fehlermeldungen!