Salomon
System przetwarzania wiedzy

Technical Documentation, ver 1.0

Historia wersji

Data	Wersja	Opis	Autorzy
17-09-2005	1.0	Tech Doc	Przemysław Misiuda, Krzysztof Nadolski, Łukasz Ostatek, Dominik Seweryn

1 Wstęp

Niniejszy dokument stanowi dokumentację techniczną projektu rozbudowy platformy Salomon o obsługę drzew decyzyjnych.

1.1 Cel

Dokument ten ma na celu przybliżenie szczegółów technicznych poszczególnych pluginów realizujących zadania kolejnych etapów procesu tworzenia i prezentowania drzew decyzyjnych. Określa on dokładnie jakie parametry przyjmują pluginy, jakie są nałożone na nie ograniczenia i jak sprawdzana jest poprawność danych.

1.2 Referencje

Niniejsze opracowanie odwołuje się do założeń określonych w dokumencie:

- Software Architecture Document

2 Pluginy

Wszystkie założenia opisane w dokumencie SAD nie uległy zmianie, a opisane niżej pluginy spełniają je.

2.1 TreeDataLoader

Plugin ten zawarty jest w pakiecie `pl.capitol.tree.plugins.dataloader`. Implementuje on wszystkie interfejsy wymagane (zgodnie z założeniami twórców) przez plugin dla platformy Salomon i jest pierwszym z grupy pluginów odpowiadających za budowę drzew decyzyjnych. Jego rola sprowadza się do wczytania dostepnych tabel, po wybraniu jednej, wczytywane są jej kolumny. Gdy użytkownik dokona wyboru ról dla kolumn informacje te przekazywane są do kolejnych pluginów.

2.1.1 Opis implementacji

Klasa `pl.capitol.tree.plugins.dataloader.TreeDataLoaderSettingsPanel` – pobiera z obiektu implementującego interfejs `IDataEngine` informacje o tabelach (tablicę obiektów implementujących interfejs ITable) w aktualnej bazie danych za pomoca metody `listTable = dataEngine.getMetaData().getTables()` oraz wyświetla je w oknie "Table". Po wyborze żądanej tabeli która posłuży do budowy drzewa wczytywane są dostępne w niej kolumny za pomocą metody `listTable[nr].getColumns()`, która

zwraca tablicę obiektów implementujących IColumn. Wybrane ustawienia zwracane są metodą `getSettings()` jako obiekt implementujący ISettings.

Zwracane wymienioną wyżej metodą ustawienia odczytywane są w dalszej kolejności w klasie pl.capitol.tree.plugins.dataloader.TreeDataLoaderPlugin, i umieszczanie w obiekcie implementującym IDataSource. Jego Id zaś umieszczany jest w obiekcie implementującym IVariable – jako tak zwana zmienna środowiskowa, którą może pobrać następny plugin i na tej podstawie zbudować drzewo.

2.2 VeniTreeCreator

Plugin należy do pakietu `pl.edu.agh.capitol.veniTreeCreator` i realizuje algorytm budowy drzew decyzyjnych. Ze wzgledu, iż jest to plugin bardziej rozbudowany niż pozostałe dwa jego opis ma nieco inną formę i w sposób bardziej szczegółowy opisuje poszczególne metody realizujące konkretne kroki algorytmu.

2.2.1 Opis implementacji

Klasa `VeniTreeCreator` - Plugin tworzący drzewa decyzyjne na podstawie danych. Drzewa są tworzone na bazie struktury IDataSource a zapisywane do ITree. Zaimplementowany algorytm tworzenia drzew to ID3.

Metody:

- doJob() - Główna metoda licząca pluginu.

  Parametry:

  - salomon.platform.IDataEngine eng
  - salomon.platform.IEnvironment env
  - salomon.plugin.ISettings settings

  Zwraca: salomon.plugin.IResult

- getDefaultErrorResult() - Rezultat zwracany w przypadku wystąpienia błędu

  Parametry:

  - java.lang.String result

  Zwraca: salomon.plugin.IResult

- getIDataSourceFromEnv() - Metoda pomocnicza pobierająca IDataSource'a z IEnvironment'u

  Parametry:

- salomon.platform.IDataEngine eng
- salomon.platform.IEnvironment env

Zwraca: salomon.platform.data.tree.IDataSource

- getResultComponent() - Zwraca komponent z rezultatem

  Parametry: Brak

  Zwraca: salomon.plugin.IResultComponent

- getSettingComponent() - Zwraca komponent ustawień

  Parametry: Brak

  Zwraca: salomon.plugin.ISettingComponent

- performCalculations() - Metoda uruchamiające logikę tworzenia drzewa

  Parametry:

  - salomon.platform.data.tree.IDataSource ds
  - salomon.platform.IDataEngine eng

  Zwraca: salomon.platform.data.tree.ITree

Klasa `DataItem` - Klasa będąca abstrakcją elementu danych używanych do tworzenia drzewa - na wewnętrzne potrzeby algorytmu ID3.

Metody:

- getAttributeAt() - Ustawia atrybuty DataItem o numerze indeks

  Parametry:

  - index int

  Zwraca: java.lang.String

- getAttributes() - Pobiera atrybuty DataItem

  Parametry: Brak

  Zwraca: java.util.Vector<java.lang.String>

- print() - Wypisuje daną zmienną

  Parametry: Nazwa zmiennej

  Zwraca: void

- pushAttribute() - Ustawia atrybuty DataItem na ostatniej pozycji

  Parametry:

  - java.lang.String attrib

  Zwraca: void

- setName() - Ustawia nazwę DataItem'u

  Parametry:

  - java.lang.String name

  Zwraca: void

- setObjective() - Ustawia wartość zmiennej decyzyjnej

  Parametry:

  - java.lang.String objective

  Zwraca: void

Klasa `TreeConstructionTask` - Klasa będąca abstrakcją zadania tworzenia drzewa decyzyjnego.

Metody:

- anyAvailable() - Test logiczny czy jest możliwe jeszcze rozwijanie względem jakiegokolwiek atrybutu.

  Parametry: Brak

  Zwraca: boolean

- calculateAverageEntropy() - Metoda matematyczna obliczająca średnią entropię w drzewie wzgledem zadanego atrybutu

  Parametry:

  - java.util.Vector<TreeItem> vt
  - int attribute

  Zwraca: double

- createTree() - Metoda inicjalizująca tworzenie drzewa

  Parametry: Brak

  Zwraca: void

- expandTree() - Metoda dokonująca rozwinięcia wg zadanego atrybutu

  Parametry:

  - int attribute

  Zwraca: void

- getBestAvailableAttribute() - Metoda wybierająca najlepszy dostępny atrybut (ten który minimalizuje entropię)

  Parametry: Brak

  Zwraca: int

- getDistinctClasses() - Metoda pomocnicza dokonująca ekstrakcji rozłącznych klas atrybutów z danych

  Parametry:

  - int attributeIndex

  Zwraca: java.util.Vector<java.lang.String>

- getDistinctClassesFromObjectives() - Metoda pomocnicza dokonująca ekstrakcji rozłącznych klas wartości z danych

  Parametry: Brak

  Zwraca: java.util.Vector<java.lang.String>

- getLeafs() - Metoda pomocnicza zwracająca wszystkie liście drzewa

  Parametry: Brak

  Zwraca: java.util.Vector<TreeItem>

- getValueOfProp() - Pobiera wartość atrybutu prop dla elementu o id id

  Parametry:

  - java.lang.String id
  - java.lang.String prop

  Zwraca: java.lang.String

- isAllHomogenous() - Pomocnicza metoda sprawdzająca czy wszystkie węzły są homogeniczne

  Parametry: Brak

  Zwraca: boolean

- loadFromDataSource() - Inicjalizuje zadanie tworzenia drzewa danymi z IDataSource'u

  Parametry:

  - salomon.platform.data.tree.IDataSource ds
  - java.util.List<java.lang.Object[]> data

  Zwraca: void

- loadFromFile() - Deprecated

  Parametry:

  - java.lang.String filename
  - int objectiveIndex

  Zwraca: void

- print() - Metoda wypisująca dzewo decyzyjne

  Parametry: Brak

  Zwraca: void

- printLeavesOnly() - Metoda wypisująca dzewo decyzyjne (tylko liście)

  Parametry: Brak

  Zwraca: void

- returnResult() - Zwraca rezultat obliczeń w postaci obiektu implementującego interfejs ITree

  Parametry:

  - salomon.platform.data.tree.ITreeManager iTreeManager
  - salomon.platform.data.tree.IDataSource ds

  Zwraca: salomon.platform.data.tree.ITree

- splitBy() - Metoda pomocnicza dokonująca rozwinięcia pojedynczego elementu drzewa decyzyjnego

  Parametry:

  - TreeItem ti
  - int attribute

Zwraca: java.util.Vector<TreeItem>

- splitBy() - Metoda pomocnicza dokonuje rozwinięcia drzewa wg zadanego atrybutu

  Parametry:

  - java.util.Vector<TreeItem> vt
  - int attribute

  Zwraca: java.util.Vector<TreeItem>

Klasa `TreeItem` - Klasa reprezentująca węzeł drzewa decyzyjnego
Metody:

- addToRoadMap() - Dodaje do wektora atrybutów nowy atrybut

  Parametry:

  - java.lang.String roadMap

  Zwraca: void

- calculateEntropy() - Metoda pomocnicza obliczająca entropię węzła

  Parametry:

  - java.util.Vector<java.lang.String> classes

  Zwraca: double

- draw() - Deprecated. Metoda pomocnicza "rysująca" węzeł

  Parametry:

  - int wciecie

  Zwraca: void

- getElements() - pobiera elementy węzła

  Parametry: Brak

  Zwraca: java.util.Vector<DataItem>

- getParent() - Pobiera ojca

  Parametry: Brak

  Zwraca: TreeItem

- getRoadMap() - Pobiera ścieżkę do ROOTa

  Parametry: Brak

  Zwraca: java.util.Vector<java.lang.String>

- isHomogenous() - Test logiczny czy wszystkie elemeny węzła są homogeniczne

  Parametry: Brak

  Zwraca: boolean

- isLeaf() - Test logiczny czy węzeł jest liściem

  Parametry: Brak

  Zwraca: boolean

- print() - Deprecated. Wypisuje węzeł

  Parametry: Brak

  Zwraca: void

- setLeaf() - Ustawia wartość logiczną - czy węzeł jest liściem

  Parametry:

  - boolean isLeaf

  Zwraca: void

- setParent() - Ustawia ojca

  Parametry:

  - TreeItem parent

  Zwraca: void

- setRoadMap() - Ustawia wektor atrybutów wg których następował podział w drodze do tego węzła

  Parametry:

  - java.util.Vector<java.lang.String> roadMap

  Zwraca: void

- subTreeItem() - Metoda pomocnicza zwracająca elementy poddrzewa spełniające kryteria

  Parametry:

- int attribute
- java.lang.String value

Zwraca: void

2.3 TreeVisualisation

Plugin ten zawarty jest w pakiecie pl.capitol.tree.plugins.test Implementuje on wszystkie interfejsy wymagane (zgodnie z założeniami twórców) przez plugin dla platformy Salomon.

2.3.1 Opis implementacji

Wyróżniamy następujące klasy:

- `pl.capitol.tree.plugins.test.panels.VisSettingsPanel` jest to klasa rozszerzająca `JPanel`, używa ona trzech komponentów:
  - textttChoice TreeChooser – jest to element który podczas inicjalizacji komponentu `VisSettingsPanel` wypełniony zostaje listą drzew dostępnych w Solution,
  - `Button buton` jest to element typu przycisk przy pomocy którego możemy usunąć drzewo z listy `TreeChooser`. Ma on zaimplementowaną klasę `MyActionListener implements ActionListener`, która to obsługuje usuwanie drzewa,
  - `Checkbox checkbox` to element przy pomocy którego zostaje podjęta decyzja czy plugin ma działać samodzielnie.

  W danej klasie jest zaimplementowana jeszcze jedna ważna metoda `ISettings getSettings()` która to pozwala na przekazanie ustawionych zmiennych w settings do elementu `VisPlugin`.
- `pl.capitol.tree.plugins.test.panels.VisResultsPanel` jest to klasa rozszerzająca `JScrollPanel`. Ważniejsze z jej metod to `JTextArea getResultText(String alert)`, która pozwala umieścić informacje o tym jak należy postępować aby wyświetlić drzewo w oknie Result. `drawTree(INode root, DefaultMutableTreeNode top)` jest metodą rekurencyjną rysującą poddrzewo danego elementu w komponencie JTree. Istotną metodą jest także `initialize()`, która podczas inicjalizacji panelu decyduje czy będzie wyświetlony komunikat czy też narysowane drzewo JTree w panelu Result.
- `pl.capitol.tree.plugins.test.VisPlugin` implementuje jedną istotną metodę: `IResult doJob(IDataEngine eng, IEnvironment env, ISettings settings)`

która to, pobiera ze zmiennej środowiskowej, lub też z elementu pobranego z `VisSettingsPanel` id drzewa, które ma zostać wyświetlone, po czym przekazuje ten identyfikator do `VisResultsPanel`.